**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«Национальный исследовательский университет ИТМО»**

**Факультет безопасности информационных технологий**

**Дисциплина:**

«Информационная безопасность»

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1** Выбор направления исследования и формализация объекта и предмета исследования (применительно к теме магистерской диссертации)

Упрощение работы с механизмами верификации программного обеспечения

**Выполнил:**

Магистрант гр. N4154c

Пазюк Владимир Романович

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Комаров Игорь Иванович

К.ф.- м.н., доцент

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

Санкт-Петербург

2022г

**Оглавление**

1. Выбор направления исследования 3

2. Обзор методов и средств моделирования систем 6

3. Моделирование объекта и предмета исследования 7

4. Верификация полученных вариантов формализованного описания. 8

5. Вывод 8

Список используемой литературы 9

# 1. Выбор направления исследования

**Формулировки тематики исследований, объекта, предмета, противоречия, цели, задачи исследования:**

**Тематика исследования** – механизмы верификации программного обеспечения.

**Цель работы** – упрощение работы с механизмами верификации программного обеспечения.

**Задачи исследования**

* Общий обзор механизмов валидации и верификации программного кода и обеспечения;
* изучить существующие механизмы и алгоритмы верификации программного обеспечения;
* оценить и сравнить работу и результаты разных механизмов верификации;
* выделить слабые стороны существующих механизмов;
* определить необходимые и возможные пути улучшения или реализации нового функционала методов верификации;
* реализовать соответствующий функционал путем создания и объединения методов верификации программного обеспечения;
* проверка и сравнение готового решения с уже существующими.

**Область исследования** – методы контроля качества и верификации в рамках разработки программного обеспечения (ПО).

**Объект исследования** – алгоритмы верификации программного обеспечения.

**Предмет исследования** – методы и инструменты статического анализа архитектуры ПО, а также динамической верификации, тестирование и верификационный мониторинг поддержки верификации ПО.

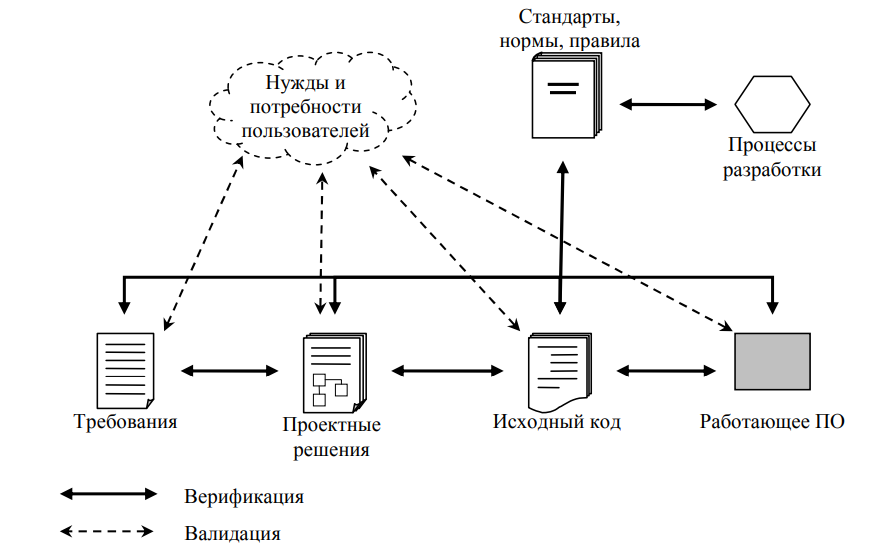
**Противоречия** – на настоящий момент не существует единого правильного способа по верификациям программного обеспечения при принятии решения о верификации программного обеспечения возникает вопрос о том, что требуется доказать, на какие вопросы способна дать ответ верификация программной системы, а именно полнота корректность, отказоустойчивость, безопасность внутренних данных, безошибочность. Так же с выходом приказа ФСТЭК России № 76 от 02.06.2020 «Об утверждении Требований по безопасности информации, устанавливающих уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий» [1] появились новые вопросы к созданию моделей для верификации. Так как нет четких требований по полноте, изолируемости и верифицируемости сегмента системы для моделирования.

**Обоснование актуальности направления:**

Вопросы об единой и правильной верификации программного обеспечения задаются уже долгое время. И только совсем недавно в связи с выходом приказа ФСТЕК России № 76 от 02.06.2020 появилась необходимость правильного и исчерпывающего алгоритма математического или программного моделирования своего решения. Существуют некоторые программы, к примеру Rodin в нотации Event-B. Которые находят «опасные» участки кода вашей модели, но нет точного определения и алгоритма для выбора уровня абстракции модели своего ПО, кроме как общие рекомендации по классическим критериям.

**Визуализация отношений между понятиями**

Для начала рассмотрим два основных понятия, а именно верификация и валидация. Верификация – подтверждение того, что определенные заранее требования реализованы в программе. Валидация – подтверждение того, что функции, обеспечиваемые системой при её использовании, соответствуют требованиям заказчика, т.е. система способна реализовывать свое предназначение.



Блок схема 1. Визуализация отношений верификаций и валидации с основными элементами в разработке ПО [2]

Вышеприведённая схема построена по содержанию стандартов программной инженерии, таком как ISO 12207

**План-график выполнения исследования**

Диаграмма 1. План-график выполнения исследования

Обзор методов валидации ПО будет содержать изучение средств и глубокое погружение в методы, которыми можно валидировать программный код, на данном этапе будут рассмотрены решения и принципы работы с ними. Будут изучены научные статьи и документация к существующем решениям.

Google scholar:

С 2018 года найдено 9000 различных результатов по запросу «валидация программного обеспечения» и 4000 результатов по «валидация программного кода».

Изучение существующих методов подразумевает работу и обзор с существующими решениями, такими как «Event-B», а также чтение новых научных статей, которых было написано согласно Google scholar с 2022 года 1210.

Оценка и сравнение методов с последующим поиском недостатков и определением решения, будет происходить с поиском лучшего решения, и просмотром отсутствующих в нем некоторых средств, реализованных в других решениях или поиск не реализованной особенности, которую можно будет добавить в готовое решение.

Реализация и проверка работы реализована в зависимости от требований к готовому решению или в виде описания метода, алгоритма для упрощения работы по валидации ПО.

Работа над диссертацией будет происходить в течении всего обучения в магистратуре.

# 2. Обзор методов и средств моделирования систем

Рассматриваются нотации: IDEF0, IDEF3, Chen, UML

**IDEF0** – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов, то есть организацию как набор модулей, здесь существует правило — наиболее важная функция находится в верхнем левом углу. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность (поток работ).

**IDEF3** – методология моделирования и стандарт документирования процессов, происходящих в системе. Метод документирования технологических процессов представляет собой механизм документирования и сбора информации о процессах, он показывает причинно-следственные связи между ситуациями и событиями в понятной эксперту форме, используя структурный метод выражения знаний о том, как функционирует система, процесс или предприятие.

**Chen** – Эта нотация была представлена Питером Ченом, являющимся одним из основоположников реляционных баз данных, и долгое время применялась для графической интерпретации предметной области в терминах сущностей и связей, иллюстрирующих ее абстрактное представление на логическом и концептуальном уровнях.

**UML –** язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования процессов, системного проектирования и отображения организационных структур.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | IDEF0 | IDEF3 | Chen | UML |
| Доступность IDE | + | + | + | + |
| Простота чтения | + | + | + | + |
| Распространенность | + | +- | - | + |
| Удобство разработки | + | - | + | + |
| Уместность в описании процессов | + | +- | + | + |
| Итог | 5 | 3 | 4 | 5 |

Таблица 1. Оценка нотаций.

Как видно из таблицы 1 были выбраны 2 нотации, а именно IDEF0, UML. Главным фактором в их выборе стала возможность описания процессов разработки ПО. В ячейках таблицы указаны положительно (+) или отрицательная (-) оценка. Чем больше положительных оценок – тем лучше.

# 3. Моделирование объекта и предмета исследования

**3.1 Функциональная модель разработки**

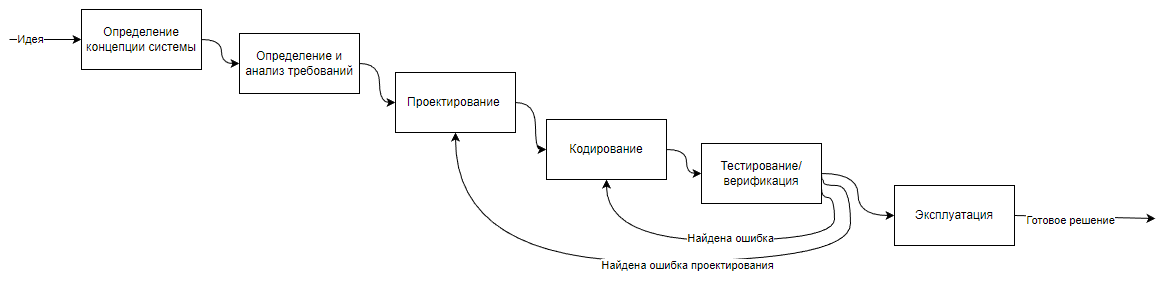
****

Рисунок 1 – IDEF0 – функциональная модель Каскадной модели жизненного цикла ПО, в которой присутствует этап верификаций ПО.

Логично будет добавить, что об этапе верификаций можно задуматься еще на этапе проектирования чтобы не допустить последующих ошибок.

**3.2 Структурная модель методов верификаций.**

Структурная модель представлена как UML – диаграмма классов

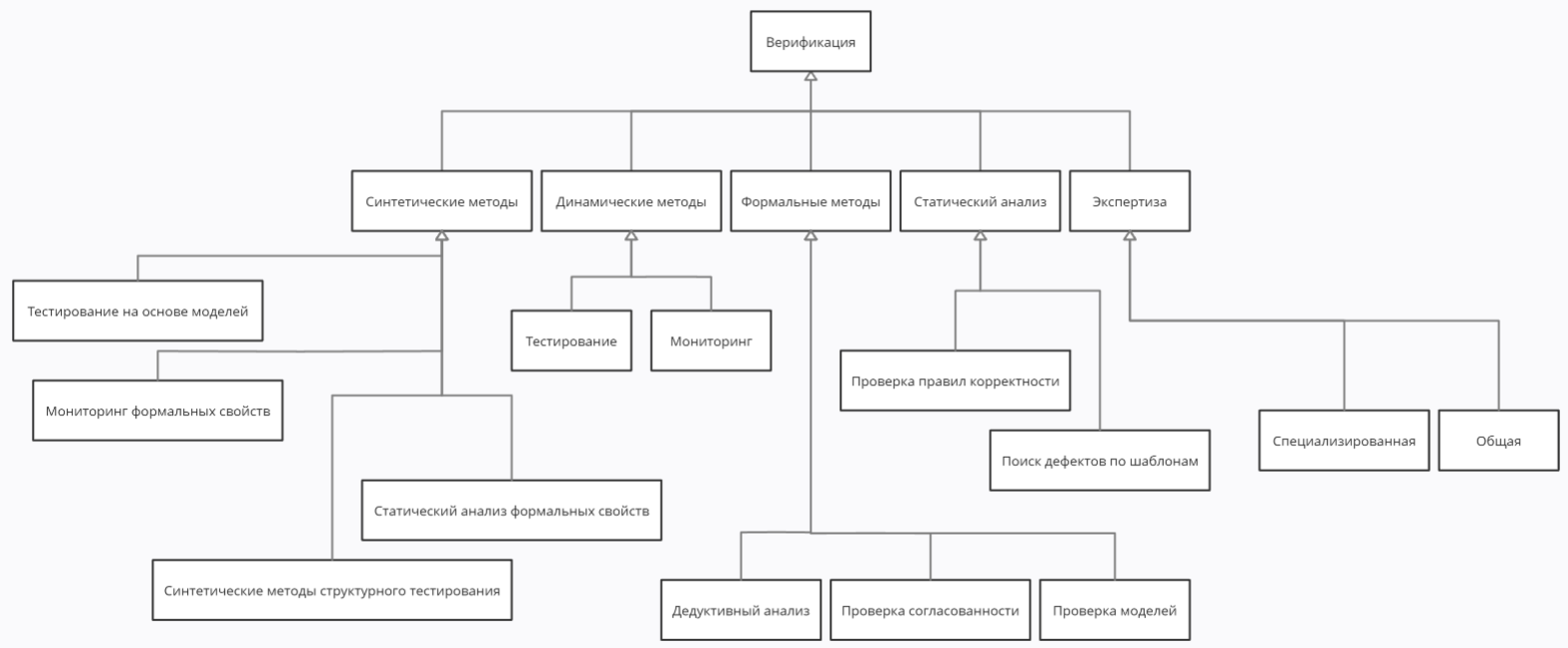
****

Рисунок 2 – UML – структурная модель, показывающая методы верификаций ПО.

**3.3 Инфологическое моделирование**

Диаграмма «сущность-связь», показывающая уязвимые места при ошибках проектирования.

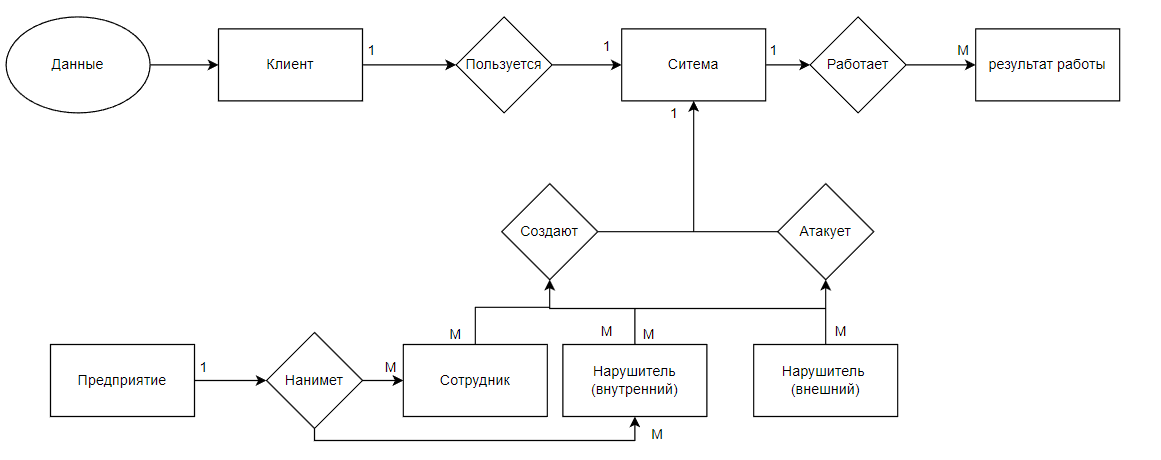


Рисунок 3 – UML – Диаграмма «сущность-связь» показывающие, что систему требуется, верифицировать начиная с этапа разработки.

# 4. Верификация полученных вариантов формализованного описания.

IDEF0 диаграмма была сделана на сайте diagrams.net и удовлетворяет всем требованиям к диаграммам IDEF0, а именно: наиболее важная функция находится в верхнем левом углу, стрелка входа всегда приходит в левую кромку активности, стрелка управления — в верхнюю кромку, стрелка механизма — нижняя кромка, стрелка выхода — правая кромка.

UML диаграммы были реализованы на сайте creately.com с учетом правил построения «диаграммы классов» UML.

# 5. Вывод

В результате проделанной работы были рассмотрены разные методы и способы верификаций программного кода, был составлен план дальнейшей исследовательской работы, и проведена работа по определению и направлению научной работы.

Поставленная тема является очень обширной и самое главное свежей в мире информационной безопасности и требует глубокого изучения. Было проведено изучение нескольких вводных научных статей, которые показали важность и место использования методов верификаций программного обеспечения, как средства защиты информации. Были поставлены следующие шаги работы и сформирован план дальнейшего развития темы научной работы.

# Список используемой литературы

1. <https://fstec.ru/tekhnicheskaya-zashchita-informatsii/dokumenty-po-sertifikatsii/120-normativnye-dokumenty/2126-vypiska-iz-trebovanij-po-bezopasnosti-informatsii-utverzhdennykh-prikazom-fstek-rossii-ot-2-iyunya-2020-g-n-76> Выписка из Требований по безопасности информации, утвержденных приказом ФСТЭК России от 2 июня 2020 г. N 76
2. <http://panda.ispras.ru/~kuliamin/docs/VerMethods-2008-ru.pdf> МЕТОДЫ ВЕРИФИКАЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ В. В. Кулямин Институт системного программирования Российской академии наук с. 7
3. ISO/IEC 27001:2013 Systems and software engineering — Software life cycle processes. Geneva, Switzerland: ISO, 2013.