Министерство образования и науки Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»

На правах рукописи

Путинцева Елена Валентиновна

# МОДЕЛИ И АЛГОРИТМЫ ТЕСТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ СПЕЦИАЛИЗИРОВАННЫХ ИСПЫТАТЕЛЬНЫХ СТЕНДОВ

2.3.3 – Автоматизация и управление технологическими процессами и производствами

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель:
д.т.н., доцент, профессор каф. КСУ
Нежметдинов Рамиль Амирович

Москва 2024

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

**Актуальность темы исследования**. На протяжении нескольких последних десятилетий область систем логического управления демонстрирует все больше и больше возможностей, достигаемых с помощью их инструментов. В соответствии с концепцией развития области микроэлектроники, сформированной Минпромторгом на период до 2030 года [97], были выявлены основные проблемы и ключевые направления развития. Одними из аспектов, требующих проработки, были озвучены технологическое отставание и существенная нехватка рынка отечественных производителей, поставляющих основные компоненты для автоматизированных систем, в том числе систем управления технологическим оборудованием. Для преодоления сложившейся ситуации предстоит масштабное увеличение производственных мощностей, перекомпоновка парка станков и станочного оборудование, а местами и полная замена имеющегося технологического оборудования новым. Соответственно, в ближайшие годы вопрос безотказной работы создаваемых систем управления станет основным аспектом эксплуатации любого оборудования или устройства. В зависимости от сложности проекта этап тестирования может занимать как 15-20% времени всего жизненного цикла (далее – ЖЦ) создания систем логического управления (далее – СЛУ), так и 40-45% для более сложных разработок. Именно поэтому становится актуальным решение вопроса проверки работоспособности проектируемых программных и программно-аппаратных систем.

На мировой арене существует множество организаций, занимающихся обеспечением безопасности работы того или иного оборудования, услуг, процессов и устройств. Зачастую подобного рода организации разрабатывают собственные сертификаты соответствия, имеющие международное значение, либо имеют возможность провести проверку на соответствие существующим мировым стандартам. Среди наиболее известных и успешно функционирующих на сегодняшний день кампаний можно назвать немецкий концерн по техническому

надзору TUV [95]. Это один из примеров того, что обеспечение безопасности работы различных систем – признаваемый во всем мире аспект эксплуатации любого оборудования или устройства. Основными видами предоставляемых услуг данной организации являются деятельность в области технического регулирования, стандартизации, метрологии, деятельность по техническому контролю, технические испытания, исследования, анализ и сертификация. Причем концерн TUV предоставляет возможность провести проверку на соответствие международным стандартам (ISO9001), а также располагает широкой базой собственных стандартов и сертификатов соответствия [96], признаваемых во всем мире. Существует множество способов решения данного вопроса, однако единое комплексное решение, имеющее в своей основе исследовательскую базу, среди отечественных научных работ обнаружено не было.

Среди рассмотренных источников [4, 58, 52, 9, 40, 125, 138, 136] имеются некоторые, так или иначе затрагивающие интересующие нас вопросы, но в них преобладающим направлением является тестирование исключительно программного обеспечения, не касаясь аппаратной части систем, зачастую сугубо практического характера применительно к узкому кругу решаемых задач (самолето- и судостроение, космическая промышленность, военная техника и др.), без исследовательского анализа и научной составляющей.

Предлагаемые в диссертации методика и алгоритмы направлены на решение вопросов безопасного использования проектируемых или модернизируемых СЛУ технологическим оборудованием (далее – ТО), а именно – процедуру тестирования. При этом предлагаемые решения имеют научно-исследовательское обоснование, позволившее выявить основные этапы жизненного цикла объектов тестирования и среды функционирования, установить их взаимосвязи, а также предложить формальное описание рассматриваемых систем.

**Степень разработанности темы исследования**. Проблемам оценки и повышения качества вновь разрабатываемых систем логического управления, круг применения и популярность использования которых растет с каждым днем, посвящены труды Р. Калбертсона, К. Брауна, Г. Кобба (Быстрое тестирование);

С. Канера, Дж. Фолка, Е.К. Нгуена (Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений); М. Фьюстера, Д. Грэхам (Software Test Automation); Г. Майерса (Искусство тестирования программ); К. Азарского (Тестирование. Легкий старт); С. Куликова «Тестирование программного обеспечения. Базовый курс».

В настоящее время существует огромное количество попыток систематизировать и классифицировать существующие способы тестирования программного обеспечения (далее – ПО) систем автоматики. С другой стороны, нет четко сформулированного и структурированного подхода к тестированию и отладке систем логического управления, включающих в себя как аппаратную, так и программную составляющие. Среди рассмотренных источников имеются некоторые, так или иначе затрагивающие интересующие нас вопросы, но в них преобладающим направлением является тестирование исключительно программного обеспечения, не касаясь аппаратной части систем, зачастую на английском языке, сугубо практического характера, изобилующие огромным количеством специфических терминов, относящихся к той или иной узкой области знаний, без научно-исследовательской составляющей и применения инструментов формализации. Поиск отечественных научных публикаций по тестированию программного или программно-аппаратного обеспечения в промышленности показал недостаточность исследований в этой области.

**Цель работы**: сокращение времени тестирования систем логического управления технологическим оборудованием за счет использования разработанных моделей и алгоритмов с применением специализированных испытательных стендов.

**Задачи исследования**. Для достижения поставленной цели в работе необходимо решить следующие научные задачи:

- Проанализировать существующие методы тестирования систем управления с учетом области их применения, выявить особенности функционирования СЛУ ТО.

- Разработать формализованное описание ЖЦ средств тестирования систем промышленной автоматики и сформировать структурную модель комплекса тестирования СЛУ.
- Разработать методику, алгоритмы и сценарии стендового тестирования СЛУ.
- Реализовать прикладные решения для тестирования СЛУ с использованием стендов.

**Объект исследования** – системы логического управления технологическим оборудованием.

**Предмет исследования** – модели, методы и алгоритмы тестирования СЛУ.

**Научная новизна**. В диссертационной работе получены следующие новые научные результаты:

1. Разработано графовое представление структуры управляемых компонент станка, позволившее выявить связи, соответствующие работе СЛУ ТО.
2. Установлены взаимосвязи между существующими видами тестирования программно-аппаратного обеспечения и их применимостью для проверки работоспособности СЛУ ТО.
3. Разработано формализованное описание ЖЦ технологического (станочного) оборудования, СЛУ ТО и стендов тестирования СЛУ, позволившее выявить объекты и процессы ЖЦ ТО, установить их взаимное влияние, а также формализовать взаимосвязи между компонентами через среду их функционирования.
4. На основе установленных взаимосвязей разработана структурная модель комплекса тестирования СЛУ, отличающаяся от известных тем, что ориентирована на возможность использования как классических (традиционных) ПЛК, так и программно реализованных контроллеров.

**Теоретическая значимость** исследования заключается в разработанной формализованной модели ЖЦ средств тестирования систем промышленной автоматики, которая может быть использована для дальнейшего развития

теоретических основ моделирования жизненных циклов объектов и процессов технических систем.

**Практическая значимость** работы заключается в:

- разработанных алгоритмах тестирования СЛУ, ориентированных на применение языка функциональных блоков, а также предусматривающих возможность тестирования программно-математического обеспечения ядра систем логического управления;
- разработанных сценариях, применяемых для ручного и автоматизированного тестирования;
- разработанной методике тестирования систем логического управления, основанной на принципе разделения их по структуре, а также позволяющей реализовывать помодульное тестирование программного обеспечения;
- реализации на базе предложенной модели стенда тестирования СЛУ и проведении тестовых испытаний.

**Методы исследования**. Теоретические исследования в диссертации базируются на основных положениях теории автоматического управления, теории графов, аппарате системного анализа, теории множеств, методах синтеза, абстракции и декомпозиции, концепции объектно-ориентированного программирования.

При решении поставленных задач использовались методы структурного анализа и моделирования, теории алгоритмов, язык программирования логических контроллеров FBD, принципы разработки человеко-машинного интерфейса.

**Положения, выносимые на защиту**:

- графовое представление структуры управляемых компонент станка;
- формализованное описание жизненных циклов технологического (станочного) оборудования, СЛУ ТО и стендов тестирования СЛУ;
- структурная модель комплекса тестирования СЛУ;
- методика тестирования СЛУ;

- алгоритмы и сценарии тестирования СЛУ, применяемые для ручного и автоматизированного тестирования.

**Степень достоверности полученных результатов** подтверждается согласованием теоретических и экспериментально полученных данных, применением системного подхода к решению поставленных задач, корректностью методов, применяемых для теоретических и экспериментальных исследований, апробацией разработанных алгоритмов и программ в системе управления электроавтоматикой металлорежущего оборудования с числовым программным управлением. Сбор, обработка, анализ и интерпретация экспериментальных данных проведена с применением математических методов и методов статистического анализа и обработки информации.

**Апробация работы**. Теоретические и практические результаты, полученные автором, докладывались на заседаниях кафедры «Компьютерные системы управления» ФГБОУ ВО «МГТУ «СТАНКИН», Международной научно-практической конференции «Наука сегодня. Вызовы, перспективы и возможности»-2019, VII Всероссийской научной конференции с международным участием «Информационные технологии и системы», Международной научно-практической конференции «Наука сегодня. Вызовы, перспективы и возможности»-2020, Всероссийской молодежной научно-технической конференции «ПРОТЭК’20».

**Соответствие диссертации паспорту специальности**. Диссертационная работа соответствует формуле научной специальности 2.3.3 – Автоматизация и управление технологическими процессами и производствами в части п. 2 – «Автоматизация контроля и испытаний.»; п. 3 – «Методология, научные основы, средства и технологии построения автоматизированных систем (АСУТП) и производствами (АСУП), а также технической подготовкой производства (АСТПП) и т. д.», п. 13 – «Методы планирования, оптимизации, модификации и эксплуатации подсистем АСУТП, АСУП, АСТПП и др., включающие задачи

управления качеством, финансами и персоналом» области исследования паспорта специальности.

**Публикации**. По теме диссертации опубликовано 16 печатных работ (из них 5 в журналах, входящих в перечень ведущих рецензируемых научных журналов и изданий, рекомендованных ВАК, 2 в журналах, индексируемых Web of Science и Scopus), включая тезисы докладов, опубликованные в рамках международных и региональных научно-технических конференций.

**Структура и объем диссертации**. Диссертационная работа состоит из введения, четырех глав, заключения, списка сокращений, списка литературы и приложений. Работа изложена на 160 страницах сквозной нумерации, включая 3 страницы списка условных сокращений и 2 страницы приложений. Содержит 32 рисунка, 19 таблиц, 146 наименований списка литературы.

# ГЛАВА 1. АНАЛИЗ СУЩЕСТВУЮЩИХ МЕТОДОВ ТЕСТИРОВАНИЯ СИСТЕМ УПРАВЛЕНИЯ С УЧЕТОМ ОБЛАСТИ ИХ ПРИМЕНЕНИЯ, ВЫЯВЛЕНИЕ ОСОБЕННОСТЕЙ ФУНКЦИОНИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ТЕХНОЛОГИЧЕСКИМ ОБОРУДОВАНИЕМ

Применение вычислительной техники в автоматическом управлении – важнейшая черта технической инфраструктуры современного общества. Промышленность, транспорт, системы связи существенно зависят от компьютерных систем управления. Практически ни одна техническая система не работает без той или иной формы управления. Яркий представитель применения систем управления – станки с числовым программным управлением (далее – ЧПУ) – прочно вошли в нашу жизнь и стали незаменимыми помощниками человека в производственной деятельности. Без этих систем было бы невозможно изготавливать многие, успевшие стать привычными и обыденными, вещи. Станки под управлением систем ЧПУ являются примером быстрого и качественного создания различных деталей и элементов, предлагая при этом изделия с недостижимой ранее точностью, а при массовом производстве – невероятно низкой себестоимостью. Следующий этап развития этой индустрии предполагает объединение отдельных станков в производственные комплексы, удешевление процесса подготовки производства и снижение стоимости самих управляющих систем.

Особенности работы систем управления сложными объектами требуют применения специализированных средств, которые, в отличие от универсальных ЭВМ, обладают функциональной компонентой, прямо ориентированной на процессы управления объектами в режиме реального времени [71]. Такими системами являются системы логического управления, реализованные на базе программируемых логических контроллеров (далее – ПЛК).

Анализ современных систем логического управления показывает, что наметился явный уход от автоматизации отдельных промышленных единиц, отдавая все большее предпочтение единым автоматизированным системам и комплексам. Следствием этого стало возникновение проблемы проверки работоспособности и отладки вновь создаваемых или подвергшихся доработке или модернизации систем управления, что возводит задачу создания алгоритмов и методик тестирования систем логического управления в ранг очевидно актуальных.

## 1.1 Обзор методов тестирования с учетом целевых областей их применения

С одной стороны, на сегодняшний день существует огромное количество попыток систематизировать и классифицировать существующие способы тестирования программного обеспечения и аппаратной составляющей. С другой стороны, нет четко сформулированного и структурированного подхода к тестированию и отладке СЛУ, включающих в себя как аппаратную, так и программную составляющие. Существуют работы, имеющие узко прикладное значение, например, судо-, ракето- и самолетостроение, автомобильная промышленность, объекты военной индустрии. При этом зачастую подобная литература публикуется на английском языке, реже на немецком, французском. Отечественные работы, формулирующие комплексную и подробную методику тестирования СЛУ промышленным технологическим оборудованием, на сегодняшний день отсутствуют. В данной работе предпринята попытка восполнить данный пробел.

В рамках реализации решения первой задачи диссертации был проведен обзор существующих на сегодняшний день методов тестирования программных систем, который позволил выделить наиболее характерные и часто используемые виды тестирования для проверки ПО на наличие ошибок и нерегулярных ситуаций.

Виды тестирования можно классифицировать по различным почти независимым признакам. Рассмотрим содержание основных из них более подробно.

**Методы тестирования по объекту тестирования**. Данный признак фокусируется на том, *ЧТО* тестируем (объект тестирования). В зависимости от того, *ЧТО* предполагается проверить на корректность работы, тестирование бывает *функциональным* и *нефункциональным*.

*Функциональное тестирование (Functional testing)* [101] – вид исследования, направленный на проверку соответствия разработанного программного продукта требованиям заказчика с точки зрения выполняемых операций и возможных функций, а именно – *что* и насколько *полно*, *точно* и *безошибочно* система в состоянии выполнять, находится ли это в соответствии с задачами, заложенные в нее на этапе проектирования и формирования технического задания. Это один из самых важных, информативных и необходимых для любых областей применения вид проверки.

*Нефункциональное тестирование (Non functional testing)* [101] проверяет все, что не связано напрямую с исследованием действующих функциональных возможностей системы. Такими параметрами могут быть следующие:

1) *Надежность (Reliability testing)* [101] – это параметр тестирования, при проверке которого определяется, правильно ли данное программное обеспечение выполняет заданные для него функции в оговоренных для его работы условиях функционирования, при этом сохраняет стабильные рабочие характеристики в течение определенного периода времени, называемого *сроком службы*.

Особенность процесса тестирования надежности состоит в том, что неудачно завершенные тест-кейсы могут обходиться достаточно дорого по сравнению с большинством других видов тестов. Поэтому существуют некоторые правила, которые необходимо соблюдать при его реализации:

- сформулировать и проинформировать всю тест-группу о целях проверки на надежность;

- составить пошаговый план тестирования;
- четко следовать предварительно составленному плану;
- на каждом шаге тестирования решения о следующих действиях принимать с учетом всех предыдущих промежуточных результатов.

Во избежание привнесения в логику работы разрабатываемой системы изменений, приводящих к полной ее неработоспособности, используются дополнительные этапы подготовки: моделирование, измерение результатов и улучшение.

2) *Защищенность (Security testing)* [101] – параметр, отслеживание которого позволяет проверить наличие и эффективность защитных механизмов, использованных в системе, и правильность их срабатывания при попытке проникновения в нее.

Чтобы принять решение о том, насколько необходимо в той или иной ситуации проведение тестов безопасности, можно воспользоваться следующим правилом: если в системе имеется ресурс с данными, несанкционированный доступ к которым грозит критическими последствиями для работы всей системы, и при этом он доступен из Интернета, то тест безопасности необходим

3) *Тестирование конфигурации (Configuration testing)* [101] имеет своей целью проверить корректность работы всех возможных конфигураций программного продукта.

Реализация конфигурационного тестирования хорошо поддается процессу автоматизации, однако для всесторонней проверки подобного рода требуется большой объем подготовленных тестовых сценариев, а в случае добавления новых возможных конфигураций проверяемой системы число тест-кейсов будет нелинейно расти.

4) *Корректность проведения процедуры инсталляции (Installation testing)* [101] – параметр, который проверяет правильность выполнения процесса установки данного программного средства на компьютер, а также возможность его обновления и удаления.

Такого характера проверка наиболее целесообразна для программных продуктов, которые предполагается эксплуатировать без ощутимых задержек и перерывов, а в случае простоя следствием могут стать значительные финансовые потери компании-разработчика либо потеря доверия заказчиков к его программному обеспечению. Такая ситуация характерна для финансовой сферы, различных банков и инвестиционных компаний (в качестве примера можно привести взаимодействие видеоблогеров, авторов различных информационно-развлекательных каналов с рекламодателями, когда человек, размещающий свои собственные видео различной тематики вставляет с некоторой периодичностью в свои видео рекламу неких продуктов или услуг, а за это получает возможность рекламировать свой блог/канал на сайтах, сотрудничающих с сайтом рекламодателя)

Работа инсталляционного ПО связана с некоторыми характерными особенностями, свойственными только ему. Это обуславливается обязательным обращением к операционной системе, а также некоторыми процедурами, выполняемыми в тесной кооперации с ней (работа с реестром, файловой системой и т.п.). В связи с этим инсталляционное тестирование предусматривает выполнение шагов, выделяющих его среди других видов проверок.

5) *Параметр корректности работы локализованных версий программного продукта (Localization testing)* [101], имеющий целью проверить правильность работы многоязыкового ПО во всех языковых версиях, предусмотренных разработчиком.

Проверки подобного рода выполняются для программных средств, которые планируется эксплуатировать в различных странах мира, либо которые рассчитаны на международную аудиторию. Создание версии продукта на различных языках подразумевает не только трансляцию всех меню и подменю, горячих клавиш, иконок, системных сообщений, ошибок, разделов «Справка»/«Помощь» и пр. на соответствующие дополнительные языки. Преобразования должны также коснуться таких аспектов интерфейса, как отображение даты и времени, единиц измерения (при необходимости), денежные единицы, принятые в той или иной

стране, возможные культурные и лингвистические особенности и пр. Идеально выполненная локальная языковая версия программного продукта создает ощущение абсолютной уверенности пользователей данного языкового сегмента в том, что используемое ПО является для них отечественным, т.е. было изначально создано разработчиками их родной страны.

6) *Удобство использования (Usability testing, юзабилити-тестирование)* [101] – параметр, направленный, как ясно из названия, на исследование удобства работы пользователя с приложением.

«Юзабилити-тестирование относится к пользовательскому опыту и проверяет, насколько приложение легко в использовании и интуитивно понятно. Такое тестирование наиболее актуально для тех интерфейсов, которые содержат в себе большое количество графических многоцветных компонентов, аудио- и видео-элементов» [112].

*7) Переносимость (Portability testing)* [101], этот параметр еще называют *портируемостью* или *кроссплатформенностью*, т.е. возможность работы приложения на различных платформах.

Анализ данного критерия дает возможность оценить, насколько просто или сложно тестируемый продукт переносится с одной программной среды, операционной системы, оборудования и пр. на другую. Степень простоты или сложности чаще всего измеряются стоимостью всех средств и процедур для адаптации программного приложения к работе в новой среде, либо фиксируется невозможность выполнения подобного переноса.

Общие рекомендации к тестированию переносимости заключаются в том, что его следует проводить, если программная система предназначена для перехода с одной аппаратной платформы, операционной системы или веб-браузера на другую (например, когда несколько подсистем совместно используют компоненты более крупной системы).

8) *Производительность (Performance testing)* [101] – т.е. способность системы сохранять корректность выполнения заложенных в нее функциональностей при воздействии больших нагрузок.

Тестирование производительности системы может быть нескольких видов, в зависимости от того, что ставится во главу угла проверки, т.е. какая задача выделяется наиболее значимой для осуществления проверки ее выполнимости.

Дадим определение наиболее часто используемым видам тестирования производительности:

- *нагрузочное тестирование (Load Testing)* [101] – это процедура, дающая возможность определить максимально возможное количество одноплановых задач, которые приложение в состоянии обрабатывать, не выходя из строя. Основную задачу нагрузочного тестирования можно кратко сформулировать в виде вопроса: «Достаточно ли быстро работает система?» Нагрузочное тестирование рекомендуется проводить при выпуске нового программного обеспечения, доработке эксплуатируемого ПО и при изменении конфигурации.
- *объемное тестирование (Volume Testing)* [101] включает в себя определение времени реакции программного продукта. Такая проверка наиболее характерна для приложений, являющихся, по своей сути, базой данных (БД). Отслеживаемыми в ходе проверки параметрами обычно являются: продолжительность операций, направленных на обращения к базе данных (выгрузка и редактирование данных); построение кривой зависимости скорости выполнения запросов к БД от количества хранимых данных; максимальное количество одновременно обрабатываемых запросов приложением без его ощутимых зависаний и сбоев в работе.
- *тестирование масштабируемости (Scalability Testing)* [101] – проверка программного продукта на предмет того, как он реагирует на возможное изменение (увеличение или уменьшение) масштабов некоторых нефункциональных параметров (количество пользователей, запросов, объема данных, частоты подсчета транзакций и т. п.).

Основная цель тестирования масштабируемости – определить лимит пользователей программного продукта. Такая проверка чаще всего применяется при тестировании веб-приложений, дабы гарантировать, что работа конечного

пользователя при высокой нагрузке на сайт не будет нарушена. Одним из примеров является возможность своевременного доступа к веб-странице с ограниченной задержкой ответа. Другой частой целью бывает проверка сервера, сможет ли он справиться, т.е. не произойдет ли сбой сервера, если он находится под большой нагрузкой [112].

- *стрессовое тестирование (Stress Testing)* [101] – проверяет способность системы сохранять работоспособность в условиях за пределами нормального (номинального) функционирования.

Задача стресс-теста – оценить устойчивость и выносливость приложения, готовность выдержать (хотя бы кратковременно) неожиданный всплеск активности. Данный вид проверки наиболее характерен при испытаниях веб-приложений.

**Методы тестирования по исполнению кода**. Исполнение кода при проведении процесса тестирования может быть статическим – *статическое тестирование (Static Testing)* [101] и динамическим – *динамическое тестирование (Dynamic Testing)* [101].

Статическое тестирование есть фактически тестирование «за столом». Для статического тестирования характерна реализация на достаточно ранних этапах разработки программного продукта, так как на более поздних этапах такая задача становится практически невыполнимой из-за значительно разросшейся и усложнившейся структуры кода.

Для проведения подобных проверок обычно привлекается специалист высокой квалификации, имеющий достаточно большой опыт в изучении текстов программ. Данный факт делает этот метод весьма затруднительным в некоторых случаях, так как наличие подобного рода специалиста не всегда является неотъемлемой частью проектной группы. Статическое тестирование может производиться как вручную, так и с помощью различных программных средств и инструментов. При этом необходимо отметить, что данный вид тестирования редко и очень плохо поддается процедуре автоматизации. Его выполнение заключается в

кропотливом и внимательном изучении текстов программ, причем важное значение для эффективности проверки имеет квалификация и опыт проверяющего.

*Динамическое тестирование* подразумевает запуск и просмотр в действии проверяемого программного продукта. Причем это процесс может проводиться тремя различными способами:

- *тестирование методом «белого ящика» (White-box Testing)* [101] – вид динамического тестирования, при котором тщательно исследуется и анализируется внутренняя структура программы, ее архитектура, изучаются все логические маршруты, цепочки и ветви кода, вводимые переменные и используемые операторы языка программирования. Очевидно, что такой подход требует доступа к исходному коду программы.

Такая задача будет по силам лишь специалистам, имеющим нетривиальные знания в различных языках и средах программирования, способах безопасного кодирования, имеющим представление об основах проектирования и владеющим знаниями о наиболее часто используемых архитектурах создания программного кода. Все эти понятия накладывают требования достаточной высокой квалификации к человеку, осуществляющему проверку системы данным способом.

- *тестирование методом «черного ящика» (Black-box Testing)* [101] рассматривает программный продукт как некий модуль (блок) с непрозрачными стенками, то есть отсутствует информация как непосредственно об исходном коде, так и о его структуре, и алгоритме работы. При проведении процедуры тестирования ориентируются исключительно на реакцию выходов в ответ на подаваемые соответствующие входные сигналы.

Несомненным преимуществом данного метода является то, что доступа к исходному коду программы не требуется. В данном варианте тестирования к исполнителю не предъявляются слишком высокие требования к знанию или владению теми или иными языками программирования и основам их использования. Вся его деятельность сводится к вводу на входные элементы системы входных данных, предусмотренных тестовых сценарием, фиксации

результатов работы логики программы и сравнению их с заданными. Зачастую, такие процедуры хорошо поддаются процессу автоматизации, и в таком случае оператору необходимо лишь одним действием инициировать начало процесса тестирования.

- *тестирование методом «серого ящика» (Gray-box Testing)* [101] является гибридным методом двух выше описанных, собрав в себе основные их преимущества: не требуется подробная информация о внутренней организации и архитектуре программы, достаточно частичное понимание; данные знания комбинируются со сведениями, полученными из технической документации об объекте тестирования, спецификаций и других сопроводительных документов, описывающих требуемое поведение системы.

При таком подходе для разработки тестовых случаев изучаются коды модулей по технике «белого», а фактическое испытание выполняется на интерфейсах программы по технологии «черного» ящиков.

**Методы тестирования по позитивности сценария**. В зависимости от сценария тестирование может быть позитивным или негативным.

При *позитивном тестировании (Positive Testing)* [101] проверяют способность программы выполнять заложенные в нее функциональные возможности, в нормальных (штатных, ожидаемых) для ПО условиях работы, при этом не должно возникать никаких сложностей. Позитивное тестирование очень похоже на функциональное и является его частным случаем. Поэтому, если ранее было проведено успешно завершившееся функциональное тестирование, то это даст нам большую вероятность отсутствия сбоев при позитивном тестировании.

*Негативное тестирование (Negative Testing)* [101] происходит на сценариях, соответствующих нештатному поведению программы, проверяется корректность работы программы в экстренных ситуациях. Это могут быть исключительные ситуации или неверные данные. Например, если мы решим умножить на калькуляторе число 4 на букву «В», в игре посадить птицу в норку, а животное – в гнездо на дереве – это будут варианты тест-кейсов для негативного тестирования.

**Методы тестирования по хронологии выполнения**. Не последнее место в классификации видов тестирования систем логического управления занимает хронология выполнения процедуры тестирования. В зависимости от данного признака процесс исследования СЛУ может представлять собой следующие разновидности.

*Входной тест (Smoke Testing* или *Smoke Test, дымовое тестирование)* [101] – специальный тип теста «на дым» для принятия решения, готов ли компонент или система для дальнейшего детального тестирования, обычно проводится в начальной фазе тестирования. Входной тест предполагает проверку работоспособности базовых минимальных функциональностей системы, видимых как бы «с первого взгляда», не прибегая к углубленному изучению внутренних функций системы. Обычно выполняется тем же, кто и создает всю логику проектируемого программного продукта. Данную проверку необходимо проводить для того, чтобы убедиться в возможности проведения дальнейшего тестирования (основного, приемочного и т.п.). Наиболее актуально как самостоятельный вид тестирования при испытании клиент-серверного соединения, например, для проверки связи с какой-либо базой данных.

*Основное тестирование (Basic Testing)* [101] – выполнение всесторонней непосредственной проверки продукта, которое, в свою очередь, подразделяется на несколько видов.

- *Модульное тестирование (Unit Testing)* [101] – вид тестирования, когда вся проверяемая программа разбивается на небольшие блоки (модули, функции, методы), а затем уже производится испытание каждого модуля в отдельности.

Наряду с функциональным тестированием модульное является одним из наиболее фундаментальных видов тестирования, нашедшим широкое применение в различных областях индустрии автоматизации. Большинство разрабатываемых на сегодняшний день систем логического управления имеют достаточно сложную структуру и организацию, при этом легко поддаются условному разбиению на некие логически обособленные блоки (модули). Это и является одной из главных

причин такой популярности модульного тестирования среди разработчиков и наладчиков систем логического управления. Оно достаточно удобно в использовании и дает хорошие результаты по количеству обнаруживаемых в результате его применения ошибок.

- *Интеграционное тестирование (Integration Testing)* [101] – проверка всей системы на наличие ошибок и багов. Осуществляется путем сопряжения аппаратных и программных компонентов всей системы и используются для проверки корректности взаимодействия между собой компонентов, которые в совокупности представляют функциональную единицу. По сути своей является обратным по отношению к предыдущему виду тестирования, но в то же время является обязательным последующим действием после проведения успешного модульного тестирования. Проводится на этапах, близких к финальному тестированию.
- *Системное тестирование (System Testing)* [101] – тестирование программы в целом, проверка соответствия программы заявленным требованиям.

На первый взгляд может показаться, что системное и интеграционное тестирования проверяют одни и те же функциональности. Однако это не совсем верно. Системное тестирование является более общем, целостным и всевмещающим методом, по сравнению с интеграционным, который отслеживает корректность взаимодействия между собой двух или более модулей системы.

*Приемочное тестирование (Acceptance Testing)* [101] – третий вид тестирования по хронологии, характерным признаком которого является то, на каком этапе оно используется. Приемочное тестирование определяет фактический уровень готовности системы к эксплуатации конечными пользователями и проводится на основании набора тестовых сценариев, покрывающих основные бизнес-операции системы.

В отличие от, например, входного, это вид тестирования является завершающим тестом, который проводится на заключительном этапе комплекса мероприятий по тестированию системы, непосредственно перед передачей

программного продукта клиенту. Кажущееся сходство приемочного тестирования с интеграционным вполне понятно, однако в отличие от второго, использует достаточно простые тестовые сценарии, в создании которых может даже участвовать сам заказчик.

*Повторное тестирование (Retesting)* [101] – обычно проводится с целью убедится, что определенный дефект был устранен и при этом вся функциональность в целом не пострадала и работает корректно.

Как было показано в п.1.2, тестирование носит итеративный характер, повторяющийся от выпуска к выпуску программного продукта. В течение своей жизни разрабатываемая система претерпевает множество корректировок и модификаций. Повторное тестирование используется после проведения провальных тестов, обнаружения неисправностей и их устранения.

*Регрессионное тестирование (Regression Testing)* [101] направлено на обнаружение ошибок в уже протестированных участках с целью проверки продукта на ошибки, которые могли появиться в результате добавления нового участка программы или исправления других ошибок. Таким образом, регрессионное тестирование является более широким и всеобъемлющим процессом, чем, повторное тестирование.

**Методы тестирования по субъекту тестирования**. В зависимости от субъекта тестирования**,** т.е. от того, *КТО* тестирует, различают альфа- и бета-тестирование.

*Альфа-тестирование (Alpha Testing)* [101] чаще всего выполняется в лабораторной среде самими разработчиками или группой тестировщиков для проверки корректности работы предварительной версии программы. Оно проводится на ранних этапах, в конце разработки системы логического управления и ДО бета-тестирования

Данный вид тестирования в процессе его выполнения касается множества других видов тестирования. Таких как функциональное, позитивное, инсталляционное, приемочное, регрессионное, что делает его выгодным с точки зрения универсальности.

При *бета-тестировании (Beta Testing)* [101] выполняется максимально интенсивное использование почти готовой версии программного продукта с целью выявления наибольшего количества ошибок в его работе ля их последующего устранения перед окончательным выходом продукта к массовому пользователю. Наиболее применим для ПО общего пользования (рассматриваемые в работе СЛУ технологическим оборудованием являются программным продуктом, создаваемым на заказ).

Бета-тестирование выполняется реальными пользователями приложения, которые не являются сотрудниками организации, так называемой группой добровольцев из числа обычных будущих пользователей системы, в реальной среде. Бета-тестирование, также как и альфа-, включает в себя множество других видов тестирования: приемочное, функциональное, тестирование методом «черного ящика». Однако бета-тестирование требует отдельной проработки касаемо вопроса организации группы добровольцев.

**Методы тестирования по способу разработки тестовых сценариев**. Важным признаком классификации видов тестирования является метод разработки тестовых сценариев, а точнее, *ЧТО* берется в основу их создания. Они могут быть: на основе требований, по вариантам использования, на основе модели.

*Тестовые сценарии на основе требований (Requirement Testing)* [101] представляют собой просмотр программного продукта на наличие неоднозначностей требований, выведение причинно-следственных связей. В качестве источника требований могут выступать непосредственно спецификации или же какие-либо дополнительные пользовательские функциональные требования к продукту. Такой вид тестирования является наиболее широко применяемым в связи со своей очевидностью и отсутствием необходимости проведения дополнительных манипуляций для создания тест-кейсов или тест-группы. По сути своей также является разновидностью функционального тестирования [112]. Важной особенностью данного вида проверки является обязательное наличие исходной документации (спецификации, техническое задание), что не всегда возможно. При отсутствии подобного рода документов с входными данными по

выполняемой разработке написание тестовых сценариев на основе требований становится невозможным.

*Тестовые сценарии по вариантам использования (Use-cases Testing)* [101] производятся для:

- функционирования системы в нормальном режиме;
- расширенном режиме;
- в исключительных ситуациях.

Нетрудно заметить, что данный вид тестирования тесно перекликается с позитивным и негативным тестированием, а также с тестированием методом «черного ящика».

*Тестовые сценарии на основе модели (Model Testing)* [101] разрабатываются, как понятно из названия, на основе модели, отражающей структуру и поведение системы. По своей сути «это одна из техник тестирования методом «черного ящика». В непрерывной разработке большого продукта ошибка может стоить дорого, и именно поэтому использование моделей (или стендов тестирования) — один из проверенных и эффективных способов предотвратить ее как можно раньше. Предлагаемые модели должны быть в меру точны, адекватны (соответствовать реальности), универсальны (могут быть использованы неоднократно и для разных задач) и целесообразно экономичны. Основные особенности тестовых моделей в том, что их можно начинать собирать еще до фактического старта разработки, и в том, что их можно обновлять и переиспользовать при изменении системы. Таким образом, тестовая модель дает более ясное представление о системе всем участникам разработки и упрощает поддержку будущей тестовой документации» [35]. В данной работе рассматривается тестирование с помощью быстросборных специализированных стендов.

**Методы тестирования по степени формальности**. Степень формальности процесса тестирования делит все подходы к тестированию на три вида:

1) *с помощью тестов (Execute the tests)* [101],
2) *специализированные* (*Ad-hoc Testing*, *интуитивные*, *свободные*) [101] и

3) *исследовательские (Exploratory Testing)* [101].

В первом случае суть подхода заключается в четком следовании заранее спланированному тестовому сценарию. Это формализованный подход, в котором тестирование производится на основе заранее подготовленных тест-кейсов, наборов тест-кейсов и иной документации. Его отличает строгая систематизация процесса, удобство применения широкого набора выработанных за десятилетия и проверенных на практике рекомендаций. Однако здесь также обязательным условием возможности проведения данного вида тестирования является наличие этой самой документации, что на практике не всегда выполняется.

*«Специализированное (свободное) тестирование* – это тестирование без разработки тестов, без определения ожидаемых результатов, без документации. Основывается на интуиции и опыте тестировщика. Соответственно, это потребует достаточного опыта работы и большого объема знаний в сфере тестирования, а также в той области, для которой создается программный продукт. Это неформальное, импровизационное тестирование. Оно не требует никакой документации, планирования, процессов, которых следует придерживаться при выполнении тестирования. Такой способ тестирования в большинстве случаев дает большее количество заведенных отчётов об ошибках. Это обусловлено тем, что тестировщик на первых шагах приступает к тестированию основной функциональной части продукта и выполняет как позитивные, так и негативные варианты возможных сценариев» [7].

Третий способ – *исследовательское тестирование* – это выявление артефактов, появляющихся в процессе разработки программного продукта путем исследования и анализа исходных файлов, таких как документация или программный код; качество исходных файлов и их соответствие требованиям оцениваются вручную, либо с использованием вспомогательных инструментов [112]. По степени формальности предполагает менее формальный подход к процедуре исследования программного продукта, подразумевает последовательное изучение объекта тестирования либо одновременную разработку тестов и параллельное их исполнение. Такая стратегия может дать очень хороший результат

в отношении обнаружения наибольшего количества ошибок, однако при условии, что тестировщик имеет большой опыт работы в данной области и четко понимает, какой он ожидает результат на выходе. Человеку, не имеющему достаточной квалификации в данной сфере, будет тяжело применять исследовательское тестирование, велика вероятность сбиться с намеченного пути и начать отвлекаться на различные мелочи. Поэтому непрофессионалу в области отладки программных систем наиболее оптимальным решением будет работать по тестам и четко следовать заранее намеченному плану действий.

**Методы тестирования по степени автоматизации**. Важным признаком классификации является *степень автоматизации*. В зависимости от этого параметра существуют следующие методы тестирования.

*Ручное тестирование (Manual Testing)* [101] проводится без использования дополнительных программных средств для выполнения тестов и проверки результатов выполнения.

При том что ручное тестирование является наиболее примитивным из всех типов тестирования, оно помогает находить ошибки в программной системе, которые могли остаться необнаруженными с помощью многих других методов проверки. Самым главным преимуществом данного метода является то, что ручное тестирование не требует знания какого-либо инструмента тестирования. Однако наличие весомого опыта у тестировщика, работающего вручную, без использования каких-либо инструментальных сред, существенно увеличивает производительность данного метода проверки.

Ручное тестирование может включать в себя некоторые описанные выше методы. Такие как: тестирование методом «черного ящика», «белого ящика», модульное тестирование, системное, интеграционное, приемочное и некоторые другие.

*Автоматизированное тестирование (Computer-assisted Testing)* [101] – это тестирование, при котором используются программные средства для выполнения тестов и проверки результатов выполнения. Автоматизированное тестирование, несомненно, приносит пользу и экономит время и ресурсы компании.

На любом проекте важны три фактора: время, стоимость и качество. Для успешного выполнения проекта важно сократить стоимость и время с сохранением качества. Внедрять автоматизацию процесса тестирования рекомендуется в том случае, если в данные момент производится:

- «регрессионное тестирование: это первый кандидат на автоматизацию из-за регулярного запуска тестов; на долгосрочных проектах позволяет значительно сократить временные затраты» [111];
- «нагрузочное тестирование: позволяет быстрее получать результаты, экономить на мощностях и стоимости инструментов» [111];
- «тестирование локализации (если продукт регулярно обновляется, добавляется новая функциональность, то тестирование локализации выгоднее проводить в автоматическом режиме)» [111].

«Стоит отметить, что нередко наиболее выигрышным сценарием является сочетание двух подходов. При этом доля автоматических и ручных тестов будет варьироваться в зависимости от требований проекта, бюджета, сроков, в которые должно уложиться тестирование, экспертизы команды» [111].

Использование научного подхода в изучении основных приемов, свойств и наиболее характерных особенностей представленных видов тестирования позволило объединить некоторые подходы в группы по общим признакам. Результатом стала приведенная ниже интеллект-карта видов тестирования систем автоматизации (Рисунок 1.1). Следует заметить, что, предложенная структура не является единственной и не претендует на исключительное право единственно правильного и полностью исчерпывающего решения, а лишь представляет собой один из возможных способов упорядочить наиболее широко используемые методы тестирования. Почти каждая группа имеет по множеству разбиений и ветвлений, которые в той или иной мере могут пресекаться между собой. В приведенной классификации различных способов проверки работоспособности системы имеет место частичное перекрытие контуров их смыслов. То есть функциональное наполнение одного в какой-то мере может повторяться в каком-либо другом или в нескольких.


**Рисунок 1.1 – Классификация видов тестирования**

## 1.2 Структура технологического оборудования и основные свойства используемых в нем систем логического управления

Для лучшего понимания особенностей объекта тестирования была рассмотрена структура управляемых элементов станка с ЧПУ в виде ориентированного графа (Рисунок 1.2), в котором вершины – это управляемые узлы и механизмы станка, а ребра – форма их взаимодействия. Жирным выделены связи, представляющие собой работу СЛУ станка – системы электроавтоматики. Основными модулями здесь являются: шпиндельный узел, привода подач, система автоматической смены инструмента (АСИ), револьверная головка (РГ), система управления охлаждением инструмента, смазочно-охлаждающие технологические средства (далее – СОТС), система управления охлаждением станка, система удаления стружки, СЧПУ, обеспечение безопасности. Описание вершин и физический смысл связей между вершинами приведены в таблицах 1.1 и 1.2.

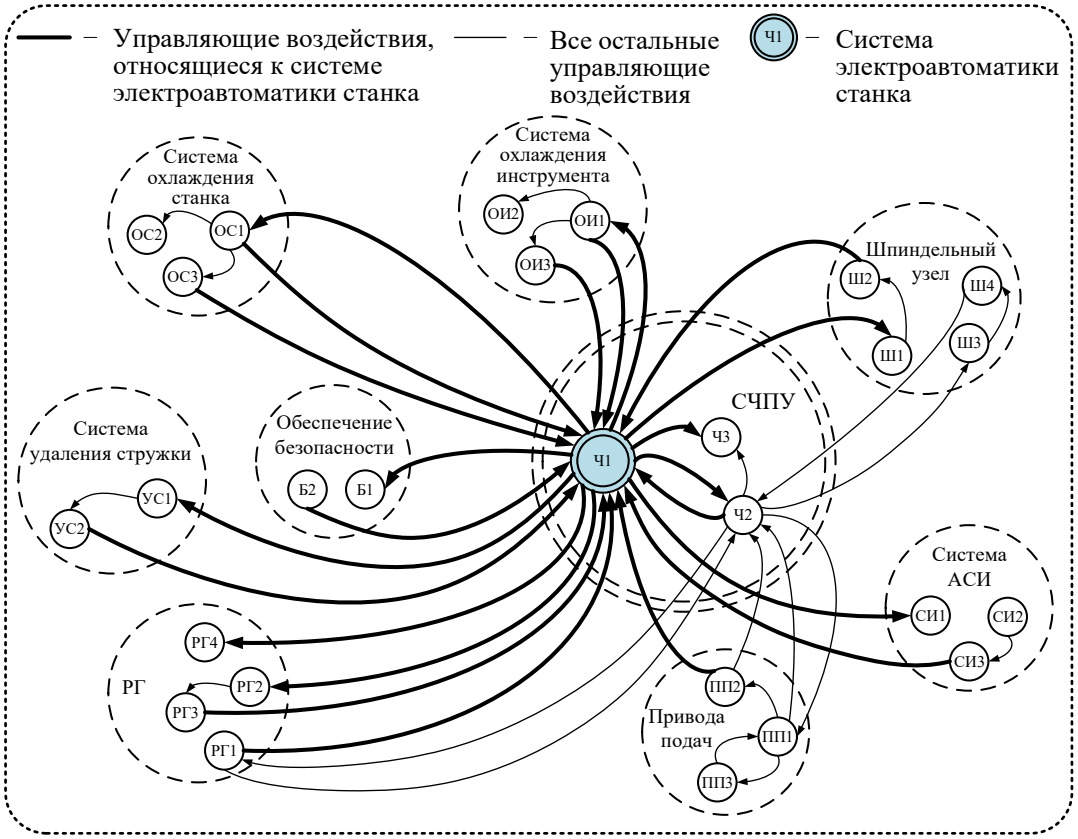



**Рисунок 1.2 – Структура управляемых компонент станка, оснащенного системой логического управления**

**Таблица 1.1 – Таблица вершин графа, описывающего структуру станка с ЧПУ**

| Функциональная группа станка | Имя вершины графа | Физический смысл |
|---|---|---|
| Шпиндельный узел | Ш1 | Привод патрона |
| | Ш2 | Датчики патрона |
| | Ш3 | Привод главного движения |
| | Ш4 | Датчик угловых перемещений |
| Привода подач | ПП1 | Сервопривод с датчиком угловых перемещений |
| | ПП2 | Ограничительные датчики |
| | ПП3 | Датчики линейных перемещений |
| Система автоматической смены инструмента (АСИ) | СИ1 | Сервопривод |
| | СИ2 | Инструментальный магазин |
| | СИ3 | Датчик позиции |
| Револьверная головка (РГ) | РГ1 | Привод с датчиком |
| | РГ2 | Механизм фиксации положения |
| | РГ3 | Датчики нулевой позиции РГ |
| | РГ4 | Предохранительная муфта |
| Система управления охлаждением инструмента (СОТС) | ОИ1 | Электронасос с фильтром и датчиком засоренности фильтра |
| | ОИ2 | Напорный клапан |
| | ОИ3 | Датчик уровня жидкости |
| Система управления охлаждением станка | ОС1 | Электронасос с фильтром и датчиком засоренности фильтра |
| | ОС2 | Напорный клапан |
| | ОС3 | Датчик уровня жидкости |
| Система удаления стружки | УС1 | Двигатель конвейера |
| | УС2 | Автоматический выключатель (контроль включения двигателя) |
| СЧПУ | Ч1 | Система электроавтоматики (ПЛК) |
| | Ч2 | Управляющая программа |
| | Ч3 | Терминальная часть |
| Обеспечение безопасности | Б1 | Привод |
| | Б2 | Датчики положения |

Ниже приведена таблица ребер графа и их физическое значение в работе технологического оборудования. Связи могут быть однонаправленными и

двунаправленными, что обозначено соответствующими стрелками в имени связи. Пара вершин, соединяемых данной связью, также указана в названии ребра графа.

**Таблица 1.2 – Таблица связей графа, описывающего структуру станка с ЧПУ**

| **Наименование связи в графе** | **Физический смысл** |
|---|---|
| *1* | *2* |
| Взаимодействие системы электроавтоматики со шпиндельным узлом (включая связи внутри узла) | |
| Ч1→Ш1 | Включение/выключение привода патрона |
| Ш2→Ч1 | Замыкание цепи обратной связи привода патрона |
| Ш1→Ш2 | Срабатывание датчика привода патрона |
| Ш3→ Ш4 | Срабатывание датчика привода главного движения |
| Взаимодействие системы электроавтоматики с приводами подач (включая связи внутри узла) | |
| ПП2→Ч1 | Отслеживание системой электроавтоматики положения по осям в «грубом нуле», фиксирование аварийной ситуации с выходом за границы резания |
| ПП1→ПП3 | Срабатывание датчиков линейных перемещений |
| ПП3→ ПП1 | Замыкание цепи обратной связи линейных перемещений |
| ПП1→ПП2 | Срабатывание ограничительных датчиков |
| Взаимодействие системы электроавтоматики с системой АСИ (включая связи внутри узла) | |
| Ч1→СИ1 | Включение/выключение сервопривода |
| СИ3→Ч1 | Замыкание цепи обратной связи по текущей позиции инструментального магазина |
| СИ2→СИ3 | Срабатывание датчика позиции инструментального магазина |
| Взаимодействие системы электроавтоматики с револьверной головкой (включая связи внутри узла) | |
| Ч1→РГ4 | Срабатывание предохранительной муфты |
| Ч1→РГ2 | Фиксация/расфиксация диска РГ |
| РГ3→Ч1 | Замыкание цепи обратной связи по положению РГ |
| РГ2→РГ3 | Срабатывание датчика положения РГ |
| РГ1→Ч1 | Замыкание цепи обратной связи от датчика привода |
| Взаимодействие системы электроавтоматики с системой управления охлаждением инструмента (включая связи внутри узла) | |
| Ч1→ОИ1 | Включение/выключение электронасоса СОЖ с предварительной проверкой засоренности фильтра |
| ОИ3→Ч1 | Замыкание цепи обратной связи по уровню СОЖ |

*Продолжение Таблицы 1.2*

| *1* | *2* |
|---|---|
| ОИ1→Ч1 | Замыкание цепи обратной связи по засоренности фильтра |
| ОИ1→ОИ3 | Срабатывание датчика уровня жидкости |
| ОИ1→ОИ2 | Подача СОЖ с помощью электронасоса |
| Взаимодействие системы электроавтоматики с системой управления охлаждением станка (включая связи внутри узла) | |
| Ч1→ОС1 | Включение/выключение электронасоса с предварительной проверкой засоренности фильтра |
| ОС3→Ч1 | Замыкание цепи обратной связи по уровню охлаждающей жидкости |
| ОС1→Ч1 | Замыкание цепи обратной связи по засоренности фильтра |
| ОС1→ОС3 | Срабатывание датчика уровня жидкости |
| ОС1→ОС2 | Подача охлаждающей жидкости с помощью электронасоса |
| Взаимодействие системы электроавтоматики с системой удаления стружки (включая связи внутри узла) | |
| Ч1→УС1 | Включение/выключение двигателя конвейера |
| УС2→Ч1 | Замыкание цепи обратной связи двигателя конвейера |
| УС1→УС2 | Срабатывание автоматического выключателя |
| Взаимодействие элементов внутри СЧПУ | |
| Ч1→Ч2 | Взаимодействие УП и системы электроавтоматики |
| Ч2→Ч1 | |
| Ч1→Ч3 | Вывод информационных сообщений от системы электроавтоматики на экран терминальной части станка |
| Ч2→Ч3 | Вывод на экран терминальной части информации о ходе процесса обработки |
| Взаимодействие системы электроавтоматики с системой обеспечения безопасности (включая связи внутри узла) | |
| Ч1→Б1 | Включение/выключение привода открытия/закрытия защитных ограждений станка |
| Б2→Ч1 | Замыкание цепи обратной связи по положению защитного ограждения |
| Прочие связи | |
| ПП1→Ч2 | Замыкание цепи обратной связи от сервопривода к СЧПУ |
| Ч2→ ПП1 | Управление скоростями подач с помощью УП СЧПУ |
| ПП2→Ч2 | Передача информации по цепи обратной связи в СЧПУ |
| РГ1→Ч2 | Замыкание цепи обратной связи от привода к СЧПУ |

*Продолжение Таблицы 1.2*

| *1* | *2* |
|---|---|
| Ч2→РГ1 | Управление приводом вращения РГ от УП СЧПУ |
| Ч2→Ш3 | Управление скоростью главного привода через УП СЧПУ |
| Ш4→Ч2 | Замыкание цепи обратной связи привода главного движения |

*Окончание Таблицы 1.2*

В данной работе будет подробно рассмотрен и изучен вопрос выбора методики и алгоритмов тестирования СЛУ технологическим оборудованием. Тестирование таких систем имеет ряд особенностей и отличий по сравнению с тестированием прикладных офисных программ, веб-сайтов, баз данных, программ, имеющих игровую направленность и т.д. Основным представителем СЛУ технологическим оборудованием является система электроавтоматики станка. Ее основными отличительными чертами можно назвать следующие:

- используется для автоматизации наиболее часто встречающихся в промышленности комбинаторных и последовательных процессов;
- в большинстве случаев представляет собой «типовую систему», объединяющую технические средства (исполнительные устройства станка) и программное обеспечение (СЧПУ);
- задачи системы электроавтоматики четко поставлены, параметры однозначно определены;
- разработка аппаратной и программной части может проводиться отдельно от оборудования, с которым он будет работать;
- программа логического управления, определяющая суть работы системы электроавтоматики, доступна для программирования и внесения корректировок неспециалисту в области информатики и предназначена для управления последовательными логическими процессами в условиях промышленной среды в «реальном масштабе времени»;

- в 90% случаев имеет место бинарный характер сигналов, следовательно, программный код представляет собой различные комбинации нескольких универсальных логических элементов;
- быстрый циклический характер работы, который определяет способность вести обработку данных в «реальном масштабе времени»;
- система устанавливается на технологическое оборудование, работа которого сопряжена с воздействием негативных факторов промышленной среды: физические и механические (температура, влажность, запыленность, вибрации, шум, удары), химические (углеводородные пары, металлическая или минеральная пыль), электрические (электростатические и электромагнитные излучения);
- пользователь подобного программного продукта (оператор станка) осуществляет лишь общий контроль за ходом технологического процесса, наблюдает за работой машин, узлов и механизмов, оценивает данные о контролируемых параметрах технологического процесса через терминал оператора;
- отсутствие развитых средств интерфейса либо сугубо лаконичное его исполнение, в большинстве случаев, типовое;
- внешнее исполнение в виде стоек или пультов/панелей оператора в виде плоскопанельных компьютеров или дисплеев;
- обычно отсутствует необходимость знать внутренние механизмы работы программы для того, чтобы предвидеть поведение системы в целом – достаточно знать соотношение вход/выход (принцип «черного ящика»).

Выявленные отличительные черты и характеристики СЛУ ТО позволили при изучении существующих методов тестирования (см. п.п. 1.1-1.2) производить выборку тех методов, которые позволят наиболее полно и при этом компактно произвести комплексную проверку работоспособности СЛУ ТО.

В ходе анализа, проведенного в рамках подготовительных работ при подготовке диссертации, были рассмотрены различные существующие

руководства по тестированию, но все они имели ряд существенных недостатков (Таблица 1.3), попытка устранить которые предпринята в данной работе.

**Таблица 1.3 – Сравнительный анализ имеющихся работ по тестированию систем автоматизации**

| Источники | Страна происхождения | Рассматриваемая область применения | Тестирование программно-аппаратной части | Комплексный научно-исследовательский подход | Формализованное описание моделей | Анализ взаимосвязей ЖЦ объектов области исследования |
|---|---|---|---|---|---|---|
| Опублико-ванные | Россия, США | Коммерческое прикладное ПО в целом | Только программной | Нет | Нет | Нет |
| Неопубли-кованные | Россия | Авиа- и самолётостроение, военная техника, судостроение, космическая промышленность | Только программной | Да | Нет | Нет |

Анализ приведенных в таблице источников показывает, что все они в той или иной мере позволяют представить многоплановость и разнообразность структуры процедуры тестирования. Работы в данной области активно ведутся и на сегодняшний день, что доказывает высокий уровень востребованности инструментов для проведения всесторонней проверки работоспособности проектируемых систем управления.

В ходе исследования было установлено, что ни в одном из анализируемых источников не уделено внимание вопросам формализации объектов области исследования и их взаимосвязей. При этом поиск отечественных научных публикаций по тестированию программного или программно-аппаратного

обеспечения в промышленности показал недостаточность исследований в этой области. В рамках данной работы собраны результаты разработок, охватывающие все описанные пробелы.

## 1.3 Систематизация аналитических данных о видах тестирования систем логического управления

Как было показано выше существует большое многообразие методов тестирования. Интеллект-карта, демонстрирующая их классификацию, приведена на рисунке 1.1. Опираясь на выделенные свойства, присущие станкам с автоматизированными системами (см. п. 1.2), составим таблицу, определяющую, на сколько тот или иной метод тестирования применим и актуален при проведении проверки на работоспособность системы логического управления технологическим оборудованием.

Подводя итоги проведенного всеобъемлющего обзора различных существующих методов тестирования, их классификации и анализа можно выделить следующие ключевые моменты и аспекты процесса тестирования:

- Автоматизированные системы охватывают практически все области и сферы нашей жизни: бизнес-системы; аппаратные решения, оснащенные ПО; практически все современные потребительские товары; военные и космические системы, информационные управляющие системы; технологическое оборудование, оснащенное системой ЧПУ и многое другое. То есть ежедневно и практически ежеминутно мы имеем дело с устройствами или системами, построенными с использованием микропроцессора. Это, в свою очередь, приводит к разработке и повсеместному появлению ПО, и, следовательно, появляется необходимость его тестирования. При этом вопрос минимизации времени внедрения программно-аппаратных устройств является главенствующим.

**Таблица 1.4 – Сопоставление методов тестирования и применимости их для СЛУ технологическим оборудованием**

| **Иерархия видов тестирования** | | | | **Особенности использования** | **Приме-нимость для СЛУ ТО** |
|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* |
| По объекту: | Функциональное | | | Всегда | Да |
| | Нефункциональное: | Тестирование локализации | | Для многоязычных веб-сайтов | Нет |
| | | Инсталляционное тестирование | | Для офисного ПО | Нет |
| | | Тестирование надежности | | Необходима дополнит подготовка моделей | Да |
| | | Тестирование портируемости | | Если предполагается перенос СУ на различные платформы (ОС, веб-браузеры) | Нет |
| | | Тестирование защищенности | | Если предполагается работа с конфиденциальными данными и возможна их утечка | Нет |
| | | Тестирование удобства использования | | Наиболее применимо к веб-сайтам и офисному ПО | Нет |
| | | Конфигурационное тестирование | | Для систем, работающих в различных конфигурациях | Нет |
| | | Тестирование производительности: | Нагрузочное | Всегда | Да |
| | | | Объемное | Характерно при работе с БД | Нет |
| | | | Тестирование масштабируемости | Характерно при работе с БД и веб-сайтами | Нет |

*Продолжение Таблицы 1.4*

| *1* | *2* | *3* | *4* | *5* | *6* |
|---|---|---|---|---|---|
| По объекту: | Нефункциональное: | Тестирование производительности: | Стрессовое тестирование | Характерно при работе с веб-сайтами | Нет |
| По субъекту: | Альфа-тестирование | | | Всегда | Да |
| | Бета-тестирование | | | Характерно для пользовательских приложений, игрового контента, веб-сайтов | Нет |
| По хронологии выполнения: | Входной тест | | | Всегда | Да |
| | Регрессионное | | | Всегда | Да |
| | Повторное | | | Всегда | Да |
| | Приемочное | | | Всегда | Да |
| | Комплексное | | | Всегда | Да |
| | Основное: | Модульное | | Для систем, имеющих выделенные блоки, модули | Да |
| | | Интеграционное | | Для систем, имеющих выделенные блоки, модули | Да |
| | | Системное | | Для систем, имеющих выделенные блоки, модули | Да |
| По степени автоматизации: | Ручное тестирование | | | Для систем, имеющих разветвленную архитектуру | Да |
| | Автоматизированное тестирование | | | Где необходимо упростить проведение рутинных операций | Да |

*Продолжение Таблицы 1.4*

| *1* | *2* | *3* | *4* | *5* | *6* |
|---|---|---|---|---|---|
| По формальности: | С помощью тестов | | | Если есть возможность их формирования | Да |
| | Исследовательское | | | При отсутствии какой-либо информации о проверяемой системе | Нет |
| | Специализированное (свободное) | | | При отсутствии какой-либо информации о проверяемой системе | Нет |
| По позитивности сценария: | Позитивное тестирование | | | Всегда | Нет |
| | Негативное тестирование | | | Всегда | Нет |
| По исполнению кода: | Статическое: | Статический анализ кода | | Всегда, но сильно увеличивает время тестирования | Нет |
| | | Рецензирование исходного кода | | Всегда, но сильно увеличивает время тестирования | Нет |
| | Динамическое: | Метод «Белого ящика» | | Для относительно несложных систем | Нет |
| | | Метод «Черного ящика» | | Всегда | Да |
| | | Метод «Серого ящика» | | Всегда | Да |
| По разработке тестовых сценариев: | На основе требований | | | Если есть документация | Да |
| | По вариантам использования | | | Если предполагаются различные варианты использования | Нет |
| | На основе модели | | | Если есть модель | Нет |

*Окончание Таблицы 1.4*

- Существуют различные подходы к выбору стратегии тестирования, учитывающей такие факторы как: имеющееся время тестирования, финансовые ограничения, критерии начала и окончания тестирования, цели тестирования, количество тестировщиков и их квалификация и многое другое. В целом можно с уверенностью сказать, что подбор подходящей концепции тестирования по сложности ничуть не уступает процессу проектирования и создания программного продукта.
- По данным нескольких источников [58, 26, 53, 84, 98, 99, 9, 40, 55, 125, 136] встречаются разнообразные подходы к процессу тестирования. В тестировании нет четких определений и правил, как в физике или математике.
- Анализ и обнаружение ошибок — это субъективный процесс. Те методы, которые решают использовать в том или ином случае, и результаты, к которым приходят, — это индивидуальные выводы. Поэтому чем разнообразней будет комплекс мер по тестированию любой разрабатываемой системы логического управления, тем всестороннее будут исследованы все ее аспекты и возможные проблемы. С другой стороны, чрезмерное разрастание тестовых сценариев может привести к значительным временным затратам.
- С учетом всех рассмотренных в п.1.1 свойств видов тестирования, а также на основе сформулированных отличительных свойств СЛУ ТО (п. 1.2), в работе предлагается использование ряда специфических признаков, позволяющих выделить методы тестирования, наиболее подходящие для систем логического управления технологическим оборудованием. Данные критерии позволяют наиболее полно и при этом компактно реализовать проверку систем логического управления. СЛУ технологическим оборудованием обладают совершенно конкретными свойствами, позволяющими выявить для них наиболее актуальные и востребованные параметры, на которые необходимо уделить особое внимание. Значительную роль в выборе таких критериев сыграло решение применения специализированных стендов тестирования. Их использование является гибким, быстросборным и наглядным инструментом для реализации проверки на

работоспособность автоматизированных систем. Соответственно и методы тестирования, отбираемые для формирования результирующего решения, должны обладать свойствами простоты, универсальности и невысокой стоимости. В то же время существует ряд признаков, которые с очень маленькой долей вероятности будут применимы в системах управления технологическим оборудованием [146, 64, 42]. Эта доля вероятности настолько мала, что в данной работе считаем возможным ею пренебречь с целью получить выигрыш во времени реализации системы, отладив ее по наиболее характерным и одновременно лаконичным признакам. Данными критериями являются:

1) сравнительно небольшая стоимость реализации,
2) быстрота,
3) простота алгоритма,
4) низкая трудоемкость реализации,
5) универсальность,
6) масштабируемость,
7) возможность агрегации,
8) информативность,
9) отсутствие необходимости привлекать специалиста высокой квалификации.

К перечисленным выше критериям применим такой термин как «полезность» для тестирования СЛУ технологическим оборудованием. В ходе анализа всех существующих методов тестирования была выявлена максимальная полезность выбранных критериев для включения того или иного метода тестирования в итоговое комплексное решение. Для большей точности поясним смысл критериев, значение которых не является очевидным и интуитивно понятным. Пункты 1-4, 9 из списка выше являются логичными к пониманию и ничем не отличаются по своему значению от привычных в повседневной жизни. Суть остальных четырех имеет смысл пояснить.

Под *универсальностью* здесь понимается необходимость проведения данного вида проверки всегда, а не только для каких-либо специфических

программных продуктов. Например, как это бывает с проверкой корректности работы локализованных версий программного продукта, имеющий целью проверить правильность работы многоязыкового ПО во всех языковых версиях, предусмотренных разработчиком.

*Масштабируемость* – свойство, при котором добавление новых блоков или функциональностей в систему не приводит к радикальным перестройкам и не парализует процесс тестирования. Примером вида тестирования, в котором это требование не выполняется, является тестирование конфигурации, когда каждое добавление новых функциональностей приводит к существенному разрастанию количества тестовых сценариев.

*Возможность агрегации* означает, что выполнение данного вида тестирования автоматически включает в себя проведение нескольких других. Например, осуществление функционального тестирования одновременно может захватить, частично или полностью, такие методы тестирования как тестирование на основе анализа матрицы требований, юзабилити тестирование, исследовательское, на основе модели.

*Информативность* – это критерий, обеспечивающий уменьшение вероятности получения провальных тестов в других видах тестирования за счет успешно пройденного данного, т.е. выполнение данного вида тестирования автоматически функционально покрывает несколько других, и таким образом его успешное выполнение увеличивает шансы на то, что и некоторые другие виды тестирования дадут положительный результат. Остальные критерии не нуждаются в уточнении.

На основе проведенных исследований и сопоставлений получаем сравнительную таблицу различных методов тестирования и комплекс методов, наиболее полно отвечающих предложенным критериям (Таблицы 1.5-1.6).

**Таблица 1.5 – Анализ видов тестирования систем логического управления**

| Метод тестирования / Критерий выбора методов | Конфигурационное / Приемочное / Переносимости | Защищенности / Инсталляционное / Локализации / Юзабилити / Объемное / Стрессовое | «Белый ящик» | Интеграционное / Альфа-тестирование | Свободное | Надежности | Исследовательское | Решение: Ручное функциональное с помощью тестовых сценариев методом «черного ящика» | Решение: Модульное | Решение: Автоматизированное функциональное с помощью тестовых сценариев методом «черного ящика» | Решение: Нагрузочное автоматизированное |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Высокая стоимость реализации* | - | - | - | - | - | + | - | - | - | - | - |
| Большие временные затраты* | - | - | +/- | +/- | - | +/- | +/- | + | - | - | +/- |
| Трудоемкость проведения* | - | +/- | + | + | - | + | + | +/- | - | - | - |
| Универсальность | - | - | - | +/- | - | +/- | - | + | +/- | +/- | + |
| Сложность реализации* | +/- | - | +/- | +/- | +/- | + | + | +/- | - | - | - |
| Масштабируемость | - | +/- | +/- | - | - | + | +/- | + | + | + | + |
| Возможность агрегации | - | - | + | +/- | +/- | - | + | + | + | + | - |
| Информативность | - | - | +/- | - | +/- | - | + | + | + | + | + |
| Наличие высококвалифицированного специалиста | +/- | +/- | + | +/- | + | +/- | + | - | +/- | - | +/- |

**Примечание*: параметр оценивается в сравнении с этим же параметром у остальных видов тестирования.

**Таблица 1.6 – Анализ видов тестирования систем логического управления с учетом коэффициентов**

| Метод тестирования / Критерий выбора методов | Конфигурационное / Приемочное / Переносимости | Защищенности / Инсталляционное /Локализации / Юзабилити / Объемное / Стрессовое | «Белый ящик» | Интеграционное / Альфа-тестирование | Свободное | Надежности | Исследовательское | Р е ш е н и е | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Ручное функциональное с помощью тестовых сценариев методом «черного ящика» | Модульное | Автоматизированное функциональное с помощью тестовых сценариев методом «черного ящика» | Нагрузочное автоматизированное |
| Высокая стоимость реализации* | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 |
| Большие временные затраты* | 2 | 2 | 1,5 | 1,5 | 2 | 1,5 | 1,5 | 1 | 2 | 2 | 1,5 |
| Трудоемкость проведения* | 2 | 1,5 | 1 | 1 | 2 | 1 | 1 | 1,5 | 2 | 2 | 2 |
| Универсальность | 1 | 1 | 1 | 1,5 | 1 | 1,5 | 1 | 2 | 1,5 | 1,5 | 2 |
| Сложность реализации* | 1,5 | 2 | 1,5 | 1,5 | 1,5 | 1 | 1 | 1,5 | 2 | 2 | 2 |
| Масштабируемость | 1 | 1,5 | 1,5 | 1 | 1 | 2 | 1,5 | 2 | 2 | 2 | 2 |
| Возможность агрегации | 1 | 1 | 2 | 1,5 | 1,5 | 1 | 2 | 2 | 2 | 2 | 1 |
| Информативность | 1 | 1 | 1,5 | 1 | 1,5 | 1 | 2 | 2 | 2 | 2 | 2 |
| Наличие высококвалифици-рованного специалиста | 1,5 | 1,5 | 1 | 1,5 | 1 | 1,5 | 1 | 2 | 1,5 | 2 | 1,5 |
| **Итого (сумма коэффициентов):** | **13** | **13,5** | **13** | **12,5** | **13,5** | **11,5** | **13** | **16** | **17** | **17,5** | **16** |

Для составления таблицы были проанализированы все вышеописанные методы тестирования по предложенным критериям. Знаком «+» обозначалось соответствие указанному критерию, знаком «-» – отсутствие соответствующего свойства, а знаком «+/-» – частичная принадлежность метода к данному признаку, когда нельзя однозначно зафиксировать его отсутствие или наличие.

Получив такую сводную таблицу, преобразуем ее в более математический вариант для однозначного выделения методов, наиболее полно соответствующих всем выбранным критериям и формирования итогового решения, включенного в комплексную методику (Таблица 1.6). Данная таблица была получена путем перевода словесных характеристик качества («+», «-», «+/-») в числовую форму с целью неоспоримости преимущества методов, вошедших в итоговое решение. Если рассматриваемый критерий положительно влияет на полноту и компактность итогового решения, то ему присваивается коэффициент 2, если критерий имеет негативное влияние – коэффициент 1, а если рассматриваемый критерий удовлетворяется в оцениваемом методе частично, ему присваивается коэффициент 1,5. Подобного рода трансформация дает тождественную таблицу, где абсолютные величины коэффициентов сами по себе не важны, так как они сравнивают методы между собой и назначаются всем видам тестирования равноправно и идентично для различных критериев. Данный подход позволяет определить в списке методов те виды тестирования, которые в сумме наберут наибольшее число баллов. Это и даст итоговое решение для включения в методику тестирования.

Таким образом результирующий вариант рассматриваемых далее в работе методов тестирования включает в себя следующие: ручное функциональное с помощью тестовых сценариев методом «черного ящика», автоматизированное функциональное с помощью тестовых сценариев методом «черного ящика», нагрузочное автоматизированное тестирование, модульное.

## 1.4 Выводы к главе 1

1. Существует огромное многообразие видов тестирования программных систем, каждый из которых обладает присущей ему областью применения, степенью трудоемкости, информативности, универсальности или специфичности. Составление интеллект-карты с выделением общих и отличительных признаков является наиболее удобной формой представления о разнообразии и взаимосвязях видов тестирования.

2. Система электроавтоматики промышленного оборудования является комплексом управляемых компонент, имеющих существенное значение во всей структуре технологического оборудования, обладающим отличительными свойствами по сравнению с другим программным обеспечением.

3. Для отбора методов тестирования, позволяющих выполнять проверку работоспособности СЛУ ТО наиболее полно и компактно, необходимо сформулировать ряд специфических признаков по итогам проведенного обзора методов тестирования.

4. Установление взаимосвязей между рассмотренными методами тестирования и типом систем логического управления является основой для формирования результирующего набора методов, имеющих ключевое значение для проведения комплексной проверки работоспособности систем логического управления технологическим оборудованием

# ГЛАВА 2. РАЗРАБОТКА ФОРМАЛИЗОВАННОГО ОПИСАНИЯ ЖИЗНЕННОГО ЦИКЛА СРЕДСТВ ТЕСТИРОВАНИЯ СИСТЕМ ПРОМЫШЛЕННОЙ АВТОМАТИКИ И ФОРМИРОВАНИЕ СТРУКТУРНОЙ МОДЕЛИ КОМПЛЕКСА ТЕСТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

## 2.1 Применение специализированных стендов тестирования для проверки работоспособности технологического оборудования

В последнее время мир стремительно меняется. На рынке программных продуктов, как и во всех других областях, происходит много изменений. И не всегда есть возможность обсудить правильность выбора того или иного решения, понять, какие инструменты лучше подойдут в данном случае, а какие хуже. «Проблема полной комплексной отладки и тестирования систем логического управления связана с индивидуальным подходом и заключается в сложности подключения полного набора согласованных входных/выходных сигналов реального технологического оборудования. На реальном объекте на этапах пуско-наладки и опытной эксплуатации для комплексного тестирования программ управления невозможно или нецелесообразно искусственное создание *полного* набора возможных ситуаций, в том числе и аварийных» [87]. Проведение комплексной проверки на реальном оборудовании создает риск повреждения дорогостоящего оборудования. Наиболее подходящим инструментом для решения этой проблемы является *стендовое тестирование*.

Выбранные в главе 1 способы тестирования – ручное функциональное с помощью тестовых сценариев методом «черного ящика», автоматизированное функциональное с помощью тестовых сценариев методом «черного ящика», нагрузочное автоматизированное тестирование на основе тестовых сценариев, модульное – хорошо адаптируются под стендовое тестирование, так как в наибольшей степени ориентированы на обеспечение таких параметров как:

соответствие функциональных возможностей системы документации проекта и требованиям заказчика, способность системы сохранять заданное быстродействие при максимальном количестве выполняемых однотипных задач.

В подавляющем большинстве случаев основной задачей систем логического управления является работа с дискретными сигналами. Если говорить о технологическом оборудовании, то в промышленности системы электроавтоматики станков более чем на 70% состоят из цифровых сигналов. Это позволяет использовать простые элементы при конструировании стенда (наборы кнопок, переключателей и лампочек), а также делает использование стендового тестирования наиболее целесообразным и удобным решением.

В основу создания комплексных испытательных стендов положена идея имитационного моделирования, используемого для отладки и испытания сложных систем управления в реальном масштабе времени. Экспериментальный стенд – это простой, быстро собираемый и быстро модифицируемый объект, в котором есть *постоянно работающие* и *подключаемые* элементы. Основное назначение разрабатываемого испытательного стенда сводится к имитации входных сигналов, вызывающих ответную реакцию программного обеспечения системы логического управления. Постоянно работающие блоки обеспечивают тестирование базовой функциональности отлаживаемой системы логического управления. Подключаемые элементы реализуют подстройку испытательного стенда под конкретный объект управления, а также дают возможность быстрой переналадки стенда в соответствии с изменившимися характеристиками и требованиями к объекту управления.

Наличие совершенных и современных экспериментальных средств – специализированных стендов – является обязательной предпосылкой, обеспечивающей эффективность и надежность разрабатываемых систем логического управления, а постоянное совершенствование экспериментальной базы – непременное условием прогресса в данной области.

## 2.2 Базовые положения концепции применения логических контроллеров в системах автоматизации

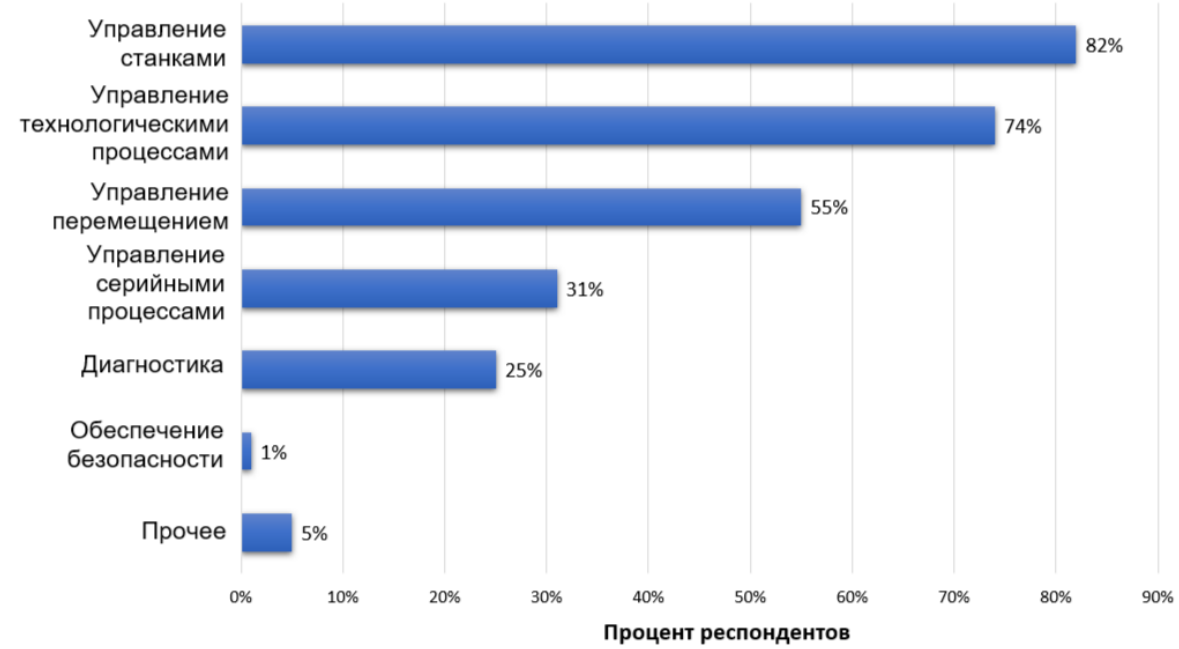
Любая система, способная автоматически выполнять какой-либо набор операций, имеет в своем составе контроллер в широком понимании данного термина (от англ. controller – регулятор, устройство управления), т.е. «модуль, обеспечивающий логику работы устройства. Другими словами, он является «мозгом» системы. Причем чем сложнее требуемая логика работы механизма, тем «умнее» должен быть контроллер» [104].

«Техническое исполнение такого контроллера может быть различным: механическое, пневматическое, гидравлическое, релейная или электронная схема, компьютерная программа» [104]. Контроллеры, выполненные на основе реле или микросхем с «жесткой» логикой, «зашитой» в их конструктивное исполнение, могут выполнять исключительно ту задачу, для которой они конструировались. Переналадить его на выполнение каких-либо других задач невозможно без существенной переделки. Поэтому популярность таких контроллеров постепенно сходит на нет. Им на смену пришли программируемые логические контроллеры, имеющие возможность взаимодействовать со средой программирования и создания различных управляющих программ. ПЛК обладают гибкостью и возможностью обработки различных программ логического управления в зависимости от решаемой задачи.

В современном мире цифровых технологий и разнообразных систем автоматизации использование программируемых логических контроллеров стало обычным и широко используемым решением (Рисунок 2.1). Такая их популярность объясняется возможностью использования в различных системах управления, покрывающих решение огромного спектра всевозможных задач (логических, технологических, системы безопасности, «умные» задачи и т.д.).

Специфика систем управления сложными объектами, какими, например, являются системы ЧПУ, требует применения специализированных компьютерных систем, которые, в отличие от универсальных ЭВМ, обладают функциональной

компонентой, прямо ориентированной на процессы управления объектами в режиме реального времени. Такими системами являются системы логического управления, реализованные на базе программируемых логических контроллеров.




*Источник*: Control Engineering и Reed Research

**Рисунок 2.1 – Области применения ПЛК**

Контроллеры, фактически, представляют собой мини-компьютер для выполнения определенных задач, под которые он запрограммирован. Предшественниками ПЛК были релейные схемы. Однако их использование делало конструкцию внешне громоздкой, функционально ограниченной и трудно обслуживаемой. Контроллеры в этом отношении предлагают более компактное, гибкое и многофункциональное решение. При этом значительно упрощается поиск неисправностей. Как правило, контроллеры используются на нижнем уровне промышленных автоматизированных систем. В состав контроллера входят: модули входов/выходов, центральный микропроцессор, микросхема памяти, интегральная схема, обеспечивающая возможность удаленного управления и обновления по одному из промышленных протоколов связи (ModBus, Hart, EtherCAT, SERCOS и т.п.), аккумулятор или батарейка на случай сбоя в электроснабжении. В зависимости от исполнения ПЛК могут иметь более или менее развитые средства

интерфейса (клавиатура, дисплей, световые индикаторы). Контроллеры обычно напрямую взаимодействуют с исполнительными механизмами и различными полевыми датчиками.

Принцип работы ПЛК заключается в считывании входных данных, поступающих на входной блок, сканировании программы логического управления и определения необходимых действий, непосредственно выполнения программы, формировании и обновлении выходных сигналов, передаче их через блок выходов, выполнении самодиагностики. Данная последовательность выполняется в цикле с определенным тактом. Время такта зависит от исполнения самого контроллера, обрабатываемых сигналов (цифровой или аналоговый), а также от используемых в нем технологий передачи данных (от нескольких миллисекунд до сотен миллисекунд).

Использование контроллеров дает ряд преимуществ перед своими предшественниками – релейными схемами:

- управляющая программа хранится в памяти самого контроллера, что означает, что в случае сбоя электроснабжения программа не обнулится, и контроллер продолжит работать;
- соединение между блоками входов и выходов происходит за счет программного обеспечения, а не с помощью прокладки сотен метров кабеля;
- компактный размер;
- возможность быстрой и несложной переналадки для решения других задач, а также более удобный и оперативный поиск неисправностей;
- возможность использования одной и той же программы управления на нескольких контроллерах (дублирование один раз написанного программного кода на различное количество устройств не займет много времени);
- возможность расширения блоков входов-выходов для увеличения количества обрабатываемых входных сигналов и количества управляемых механизмов.

По мере развития области электроники и схемотехники ПЛК приобрели тенденцию уменьшения размеров, при этом все расширяя свои функциональные возможности. Еще большее развитие данного направления дало возможность

получить виртуальное решение аппаратной части ПЛК, так называемый программно реализованный контроллер, в англоязычной литературе SoftPLC.

## 2.3 Формализованное описание процессов и объектов области исследования

Перед тем как перейти непосредственно к решению второй задачи работы проведем исследование процессов и объектов жизненного цикла (далее – ЖЦ) технологического оборудования с целью выявить и установить взаимосвязи и научно обосновать выбранную в дальнейшем структурную модель.

Результатом проведенного анализа взаимосвязей жизненных циклов объектов технологического оборудования стала схема на рисунке 2.2. На данной схеме рассмотрены и описаны основные этапы и процессы создания технологического оборудования, оснащенного системой логического управления, включая соподчиненные процессы. Красным выделены элементы, относящиеся к области исследования в рамках данной работы. Это разработка стенда СЛУ, разработка программы логического управления и конфигурация аппаратных входов/выходов, непосредственно тестирование и доработка кода и конфигурации входов/выходов. Этот этап занимает до 40% времени всего ЖЦ СЛУ и является одним из самых трудоемких, поэтому важно обратить на него серьезное внимание, учитывая, что, как было сказано ранее, работы по данной области отсутствуют.

Рассмотрим процессы жизненного цикла технологического оборудования в виде трех соподчиненных жизненных циклов:

1. ЖЦ разработки технологического оборудования, оснащенного системой логического управления;
2. ЖЦ разработки системы логического управления для технологического оборудования;

3. ЖЦ разработки испытательного стенда для тестирования созданной системы логического управления.

Каждый из трех ЖЦ включает в себя ряд процессов, последовательно или параллельно выполняемых на соответствующем этапе (Рисунок 2.2). Красной пунктирной линией, а именно – шаги 1.2.3 (1.2.3.1 и 1.2.3.2), 1.2.4 и 1.2.5 – очерчена область исследования в рамках работы над диссертацией. Связи с 1 по 4 показывает процессы, за счет которых происходит взаимодействие между тремя ЖЦ.

Переходя от графического описания к формальному, воспользуемся понятиями теории систем, которые позволяют описать систему в виде двух уравнений: взаимосвязь объектов и процессов внутри системы и взаимодействие системы с внешней средой.

Уравнение, описывающее систему без учета влияния внешних факторов, в общем виде, выглядит следующим образом:

$$S_1 =< Y, X, q(Y), q(X), R_1 > [13], \quad (2.1)$$

где

$Y$ – процессы, выполняемые описываемой системой $S_1$; $X$ – множество входных/выходных объектов, участвующих в процессах описываемой системы $S_1$; $q(Y)$ – множество разнородных свойств, характеризующих процессы системы $S_1$; $q(X)$ – множество разнородных свойств, описывающих входные/выходные объекты описываемой системы $S_1$; $R$ – множество всевозможных связей между компонентами системы $S_1$.

Уравнение с учетом взаимодействия системы $S_1$ с внешней средой в общем виде выглядит следующим образом:

$$S_2 =< S_1, Z, R_2 > [13], \quad (2.2)$$

где $Z =< SR, q(SR), R_Z >$ – это структура, включающая множество элементов среды $SR$, множество свойств элементов среды $q(SR)$ и множество связей между компонентами этой среды $R_Z$; $R_2$ – множество связей между компонентами системы $S_2$.

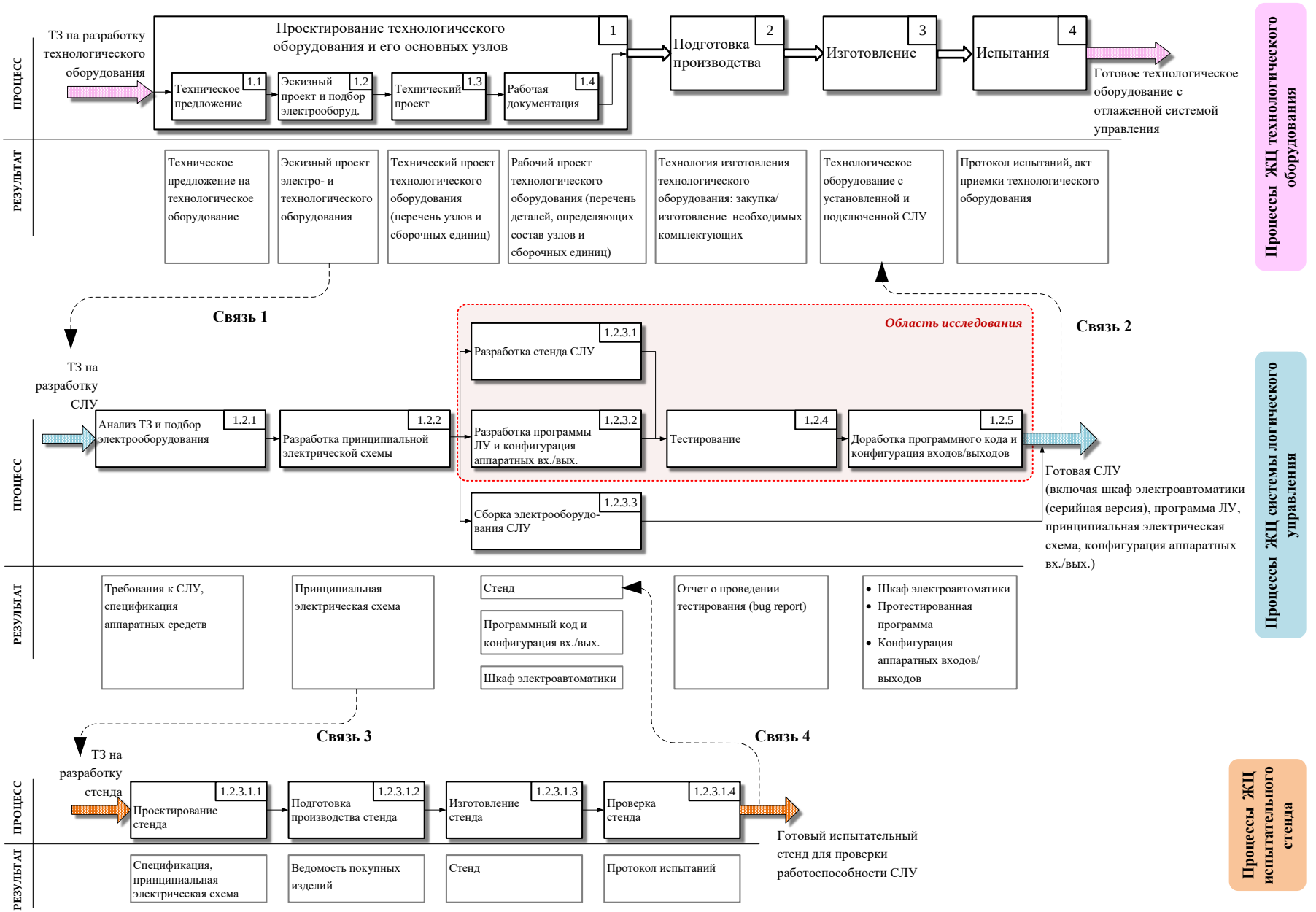



**Рисунок 2.2 – Взаимосвязи жизненных циклов различных объектов технологического оборудования и средств тестирования его СЛУ**

Для приложения представленных формальных описаний к рассматриваемым на рисунке 2.2 процессам и подпроцессам жизненного цикла технологического оборудования была введена соответствующая индексация компонентов (Таблица 2.1).

**Таблица 2.1 – Индексация компонент формального описания ЖЦ ТО**

| | | | |
|---|---|---|---|
| *n* –рассматриваемый уровень ЖЦ ТО | *EQ* (equipment) – процессы ЖЦ ТО | *k* – рассматриваемый процесс соответствующег о уровня ЖЦ ТО | *DG* (design) – проектирование технологического оборудования и его основных узлов |
| | | | *PM* (preparation of manufacture) – подготовка производства ТО |
| | | | *M* (manufacture) – изготовление ТО |
| | | | *T* (testing) – испытания |
| | *CS* (control system) – процессы ЖЦ СЛУ | *k* – рассматриваемый процесс соответствующего уровня ЖЦ ТО | *RA* (requirements analysis) – анализ ТЗ и подбор электрооборудования |
| | | | *DC* (development of circuit) – разработка принципиальной электрической схемы |
| | | | *DTS* (development of test stand) – разработка стенда системы логического управления |
| | | | *DPC* (development of program code) – разработка программы ЛУ и конфигурация аппаратных вх./вых. |
| | | | *HA* (hardware assembly) – сборка электрооборудования СЛУ |
| | | | *T* (testing) - тестирование |
| | | | *IPC* (improving the program code) – доработка программного кода и конфигурация входов/выходов |
| | *TS* (test stand) – процессы ЖЦ испытательного стенда | *k* – рассматриваемый процесс соответствующего уровня ЖЦ ТО | *DG* (design) – проектирование стенда |
| | | | *PM* (preparation of manufacture) – подготовка производства стенда |
| | | | *M* (manufacture) – изготовление стенда |
| | | | *T* (testing) – проверка стенда |

Таким образом, формальное описание процессов ЖЦ трех различных объектов в общем виде примет вид:

$$S_{k1}^{n} =< Y_k^n, X_k^n, q(Y_k^n), q(X_k^n), R_{k1}^n > [122], \quad (2.3)$$

где

$Y_k^n$ – процессы, выполняемые системой $S_{k1}^n$ в рамках ЖЦ объекта n;

$X_k^n$ – множество входных/выходных объектов, участвующих в процессах описываемой системы $S_{k1}^n$;

$q(Y_k^n)$ – множество разнородных свойств, характеризующих процессы системы $S_{k1}^n$;

$q(X_k^n)$ – множество разнородных свойств, описывающих входные/выходные объекты описываемой системы $S_{k1}^n$;

$R_{k1}^n$ – множество связей между компонентами системы $S_{k1}^n$.

Уравнение с учетом взаимодействия системы $S_{k1}^n$ с внешней средой в общем виде приобретет следующий вид:

$$S_{k2}^{n} =< S_{k1}^n, Z_k^n, R_{k2}^n > [122], \quad (2.4)$$

где

$Z_k^n =< SR_k^n, q(SR_k^n), R_{kZ}^n >$ – это структура, описывающая множество элементов $SR_k^n$, множество свойств элементов $q(SR_k^n)$ и множество связей между компонентами этой структуры $R_{kZ}^n$; $R_{k2}^n$ - множество связей между компонентами системы $S_{k2}^n$.

Применим представленные формулы (2.1) – (2.4) для описания процессов и объектов, изображенных на рисунке 2.2. Тогда, с учетом уравнения (2.3) и индексов компонент таблицы 2.1, формальные описания процессов и объектов, изображенных на рисунке 2.2, будут выглядеть следующим образом.

1. Для процессов ЖЦ технологического оборудования.

Описание системы в аспекте этапа проектирования ТО (шаг 1 на рисунке 2.2):

$$S_{DG1}^{EQ} =< Y_{DG}^{EQ}, X_{DG}^{EQ}, q\left(Y_{DG}^{EQ}\right), q\left(X_{DG}^{EQ}\right), R_{DG1}^{EQ} >, \quad (2.5)$$

где $Y_{DG}^{EQ}$ – структура процесса проектирования ТО; $X_{DG}^{EQ}$ – множество входных/выходных объектов, включающее ТЗ, техническое предложение, эскизный, технический и рабочий проекты на разработку ТО; $q\left(Y_{DG}^{EQ}\right)$ – множество различных свойств, описывающих элементы структуры процесса проектирования

ТО; $q\left(X_{DG}^{EQ}\right)$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{DG1}^{EQ}$ – множество связей между компонентами системы $S_{DG1}^{EQ}$.

Описание системы в аспекте этапа подготовки производства ТО (шаг 2 на рисунке 2.2):

$$S_{PM1}^{EQ} =< Y_{PM}^{EQ}, X_{PM}^{EQ}, q\left(Y_{PM}^{EQ}\right), q\left(X_{PM}^{EQ}\right), R_{PM1}^{EQ} >, \quad (2.6)$$

где $Y_{PM}^{EQ}$ – структура процесса подготовки производства ТО; $X_{PM}^{EQ}$ – множество входных/выходных объектов, включающее рабочий проект на разработку и технологию изготовления ТО; $q\left(Y_{PM}^{EQ}\right)$ – множество различных свойств, описывающих элементы структуры процесса подготовки производства ТО; $q\left(X_{PM}^{EQ}\right)$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{PM1}^{EQ}$ – множество связей между компонентами системы $S_{PM1}^{EQ}$.

Описание системы в аспекте этапа изготовления ТО (шаг 3 на рисунке 2.2):

$$S_{M1}^{EQ} =< Y_{M}^{EQ}, X_{M}^{EQ}, q\left(Y_{M}^{EQ}\right), q\left(X_{M}^{EQ}\right), R_{M1}^{EQ} >, \quad (2.7)$$

где $Y_{M}^{EQ}$ – структура процесса изготовления ТО; $X_{M}^{EQ}$ – множество входных/выходных объектов, включающее технологию изготовления ТО и готовое ТО с установленной и подключенной СЛУ; $q\left(Y_{M}^{EQ}\right)$ – множество различных свойств, описывающих элементы структуры процесса изготовления ТО; $q\left(X_{M}^{EQ}\right)$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{M1}^{EQ}$ – множество связей между компонентами системы $S_{M1}^{EQ}$.

Описание системы в аспекте этапа испытаний ТО (шаг 4 на рисунке 2.2):

$$S_{T1}^{EQ} =< Y_{T}^{EQ}, X_{T}^{EQ}, q\left(Y_{T}^{EQ}\right), q\left(X_{T}^{EQ}\right), R_{T1}^{EQ} >, \quad (2.8)$$

где $Y_{T}^{EQ}$ – структура процесса испытаний ТО; $X_{T}^{EQ}$ – множество входных/выходных объектов, включающее изготовленное ТО до испытаний и проверенное ТО; $q\left(Y_{T}^{EQ}\right)$ – множество различных свойств, описывающих элементы структуры процесса испытаний ТО; $q\left(X_{T}^{EQ}\right)$ – множество различных свойств,

характеризующих множество входных/выходных объектов; $R_{T1}^{EQ}$ – множество связей между компонентами системы $S_{T1}^{EQ}$.

2. Для процессов ЖЦ СЛУ ТО.

Описание системы в аспекте этапа анализ ТЗ и подбор электрооборудования для СЛУ (шаг 1.2.1 на рисунке 2.2):

$$S_{RA1}^{CS} =< Y_{RA}^{CS}, X_{RA}^{CS}, q(Y_{RA}^{CS}), q(X_{RA}^{CS}), R_{RA1}^{CS} >, \quad (2.9)$$

где $Y_{RA}^{CS}$ – структура процесса анализа ТЗ на разработку СЛУ; $X_{RA}^{CS}$ – множество входных/выходных объектов, включающее ТЗ на разработку СЛУ, список требований к СЛУ, спецификацию аппаратных средств; $q(Y_{RA}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса анализа ТЗ на СЛУ; $q(X_{RA}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{RA1}^{CS}$ – множество связей между компонентами системы $S_{RA1}^{CS}$.

Описание системы в аспекте этапа разработки принципиальной электрической схемы для СЛУ (шаг 1.2.2 на рисунке 2.2):

$$S_{DC1}^{CS} =< Y_{DC}^{CS}, X_{DC}^{CS}, q(Y_{DC}^{CS}), q(X_{DC}^{CS}), R_{DC1}^{CS} >, \quad (2.10)$$

где $Y_{DC}^{CS}$ – структура процесса разработки принципиальной электрической схемы для СЛУ; $X_{DC}^{CS}$ – множество входных/выходных объектов, включающее список требований к СЛУ, спецификацию аппаратных средств, принципиальную электрическую схему; $q(Y_{DC}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса разработки принципиальной электрической схемы; $q(X_{DC}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{DC1}^{CS}$ – множество связей между компонентами системы $S_{DC1}^{CS}$.

Описание системы в аспекте этапа разработки испытательного стенда (шаг 1.2.3.1 на рисунке 2.2):

$$S_{DTS1}^{CS} =< Y_{DTS}^{CS}, X_{DTS}^{CS}, q(Y_{DTS}^{CS}), q(X_{DTS}^{CS}), R_{DTS1}^{CS} >, \quad (2.11)$$

где $Y_{DTS}^{CS}$ – структура процесса разработки стенда; $X_{DTS}^{CS}$ – множество входных/выходных объектов, включающее принципиальную электрическую

схему, стенд; $q(Y_{DTS}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса разработки стенда; $q(X_{DTS}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{DTS1}^{CS}$ – множество связей между компонентами системы $S_{DTS1}^{CS}$.

Описание системы в аспекте этапа разработки программы логического управления и конфигурации аппаратных входов-выходов СЛУ (шаг 1.2.3.2 на рисунке 2.2):

$$S_{DPC1}^{CS} =< Y_{DPC}^{CS}, X_{DPC}^{CS}, q(Y_{DPC}^{CS}), q(X_{DPC}^{CS}), R_{DPC1}^{CS} >, \quad (2.12)$$

где $Y_{DPC}^{CS}$ – структура процесса разработки программы логического управления; $X_{DPC}^{CS}$ – множество входных/выходных объектов, включающее принципиальную электрическую схему, программный код и начальную конфигурацию аппаратных входов-выходов; $q(Y_{DPC}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса разработки программы логического управления; $q(X_{DPC}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{DPC1}^{CS}$ – множество связей между компонентами системы $S_{DPC1}^{CS}$.

Описание системы в аспекте этапа сборки электрооборудования для СЛУ (шаг 1.2.3.3 на рисунке 2.2):

$$S_{HA1}^{CS} =< Y_{HA}^{CS}, X_{HA}^{CS}, q(Y_{HA}^{CS}), q(X_{HA}^{CS}), R_{HA1}^{CS} >, \quad (2.13)$$

где $Y_{HA}^{CS}$ – структура процесса сборки электрооборудования; $X_{DG}^{EQ}$ – множество входных/выходных объектов, включающее принципиальную электрическую схему и шкаф электроавтоматики; $q(Y_{HA}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса сборки электрооборудования; $q(X_{HA}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{HA1}^{CS}$ – множество связей между компонентами системы $S_{HA1}^{CS}$.

Описание системы в аспекте этапа тестирования СЛУ (шаг 1.2.4 на рисунке 2.2):

$$S_{T1}^{CS} =< Y_{T}^{CS}, X_{T}^{CS}, q(Y_{T}^{CS}), q(X_{T}^{CS}), R_{T1}^{CS} >, \quad (2.14)$$

где $Y_T^{CS}$ – структура процесса тестирования СЛУ; $X_T^{CS}$ – множество входных/выходных объектов, включающее программный код, стенд и отчет о проведенном тестировании; $q(Y_T^{CS})$ – множество различных свойств, описывающих элементы структуры процесса тестирования СЛУ; $q(X_T^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{T1}^{CS}$ – множество связей между компонентами системы $S_{T1}^{CS}$.

Описание системы в аспекте этапа доработки программного кода и конфигурации аппаратных входов-выходов СЛУ (шаг 1.2.5 на рисунке 2.2):

$$S_{IPC1}^{CS} =< Y_{IPC}^{CS}, X_{IPC}^{CS}, q(Y_{IPC}^{CS}), q(X_{IPC}^{CS}), R_{IPC1}^{CS} >, \quad (2.15)$$

где $Y_{IPC}^{CS}$ – структура процесса доработки; $X_{IPC}^{CS}$ – множество входных/выходных объектов, включающее отчет о проведенном тестировании, готовую СЛУ, окончательную конфигурацию аппаратных входов-выходов, скорректированную принципиальную электрическую схему и шкаф электроавтоматики; $q(Y_{IPC}^{CS})$ – множество различных свойств, описывающих элементы структуры процесса доработки; $q(X_{IPC}^{CS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{IPC1}^{CS}$ – множество связей между компонентами системы $S_{IPC1}^{CS}$.

3. Для процессов ЖЦ испытательного стенда.

Описание системы в аспекте этапа проектирования стенда (шаг 1.2.3.1.1 на рисунке 2.2):

$$S_{DG1}^{TS} =< Y_{DG}^{TS}, X_{DG}^{TS}, q(Y_{DG}^{TS}), q(X_{DG}^{TS}), R_{DG1}^{TS} >, \quad (2.16)$$

где $Y_{DG}^{TS}$ – структура процесса проектирования стенда; $X_{DG}^{TS}$ – множество входных/выходных объектов, включающее ТЗ на разработку стенда, спецификацию, принципиальную электрическую схему; $q(Y_{DG}^{TS})$ – множество различных свойств, описывающих элементы структуры процесса проектирования стенда; $q(X_{DG}^{TS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{DG1}^{TS}$ – множество связей между компонентами системы $S_{DG1}^{TS}$.

Описание системы в аспекте этапа подготовки производства стенда (шаг 1.2.3.1.2 на рисунке 2.2):

$$S_{PM1}^{TS} =< Y_{PM}^{TS}, X_{PM}^{TS}, q(Y_{PM}^{TS}), q(X_{PM}^{TS}), R_{PM1}^{TS} >, \quad (2.17)$$

где $Y_{PM}^{TS}$ – структура процесса подготовки производства стенда; $X_{PM}^{TS}$ – множество входных/выходных объектов, включающее спецификацию, принципиальную электрическую схему, ведомость покупных изделий; $q(Y_{PM}^{TS})$ – множество различных свойств, описывающих элементы структуры процесса подготовки производства стенда; $q(X_{PM}^{TS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{PM1}^{TS}$ – множество связей между компонентами системы $S_{PM1}^{TS}$.

Описание системы в аспекте этапа изготовления стенда (шаг 1.2.3.1.3 на рисунке 2.2):

$$S_{M1}^{TS} =< Y_{M}^{TS}, X_{M}^{TS}, q(Y_{M}^{TS}), q(X_{M}^{TS}), R_{M1}^{TS} >, \quad (2.18)$$

где $Y_{M}^{TS}$ – структура процесса изготовления стенда; $X_{M}^{TS}$ – множество входных/выходных объектов, включающее ведомость покупных изделий, стенд; $q(Y_{M}^{TS})$ – множество различных свойств, описывающих элементы структуры процесса изготовления стенда; $q(X_{M}^{TS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{M1}^{TS}$ – множество связей между компонентами системы $S_{M1}^{TS}$.

Описание системы в аспекте этапа проверки стенда (шаг 1.2.3.1.4 на рисунке 2.2):

$$S_{T1}^{TS} =< Y_{T}^{TS}, X_{T}^{TS}, q(Y_{T}^{TS}), q(X_{T}^{TS}), R_{T1}^{TS} >, \quad (2.19)$$

где $Y_{T}^{TS}$ – структура процесса проверки стенда; $X_{T}^{TS}$ – множество входных/выходных объектов, включающее готовый стенд и протокол испытаний; $q(Y_{T}^{TS})$ – множество различных свойств, описывающих элементы структуры процесса проверки стенда; $q(X_{T}^{TS})$ – множество различных свойств, характеризующих множество входных/выходных объектов; $R_{T1}^{TS}$ – множество связей между компонентами системы $S_{T1}^{TS}$.

Представим формулы трех ЖЦ (2.5) – (2.19) в виде обобщенных.

Для описания всех подсистем ЖЦ ТО:

$$S_1^{EQ} = S_{DG1}^{EQ} \cap S_{PM1}^{EQ} \cap S_{M1}^{EQ} \cap S_{T1}^{EQ} \quad (2.20)$$

Для описания всех подсистем ЖЦ СЛУ:

$$S_1^{CS} = S_{RA1}^{CS} \cap S_{DC1}^{CS} \cap (S_{DTS1}^{CS} \cup S_{DPC1}^{CS} \cup S_{HA1}^{CS}) \cap S_{T1}^{CS} \cap S_{IPC1}^{CS} \quad (2.21)$$

Для описания всех подсистем ЖЦ испытательного стенда:

$$S_1^{TS} = S_{DG1}^{TS} \cap S_{PM1}^{TS} \cap S_{M1}^{TS} \cap S_{T1}^{TS} \quad (2.22)$$

На рисунке 2.3 представлена схема взаимосвязей всех трех жизненных циклов через формальное описание систем, процессов и объектов с использованием выражений (2.20) – (2.22). Связи 1 и 3 показывают, что результаты выполнения процесса 1.2 используются в качестве входных объектов для процесса 1.2.1, а результаты выполнения процесса 1.2.2 – в качестве входных объектов для процесса 1.2.3.1.1. Связи 2 и 4 показывают взаимодействие, осуществляемое через среду функционирования.

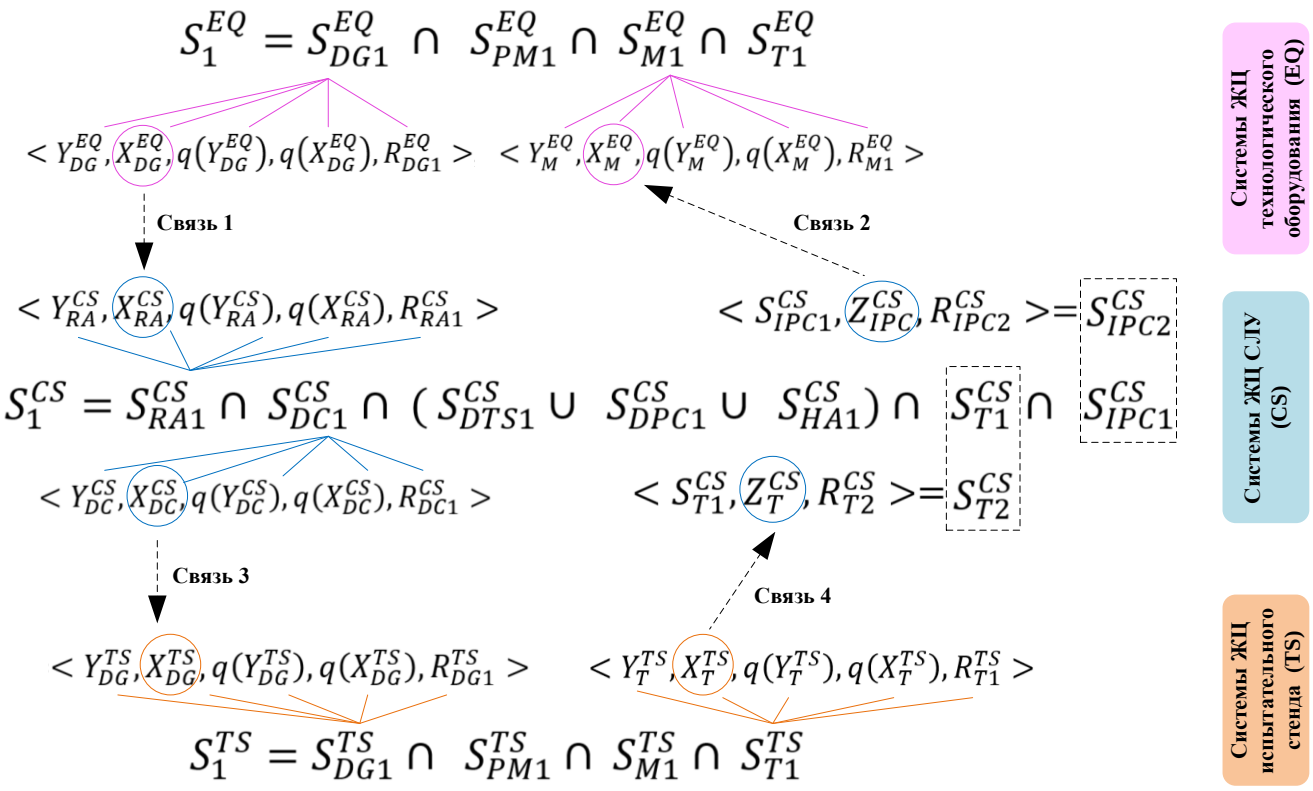



**Рисунок 2.3 – Формальное описание и связи компонентов ЖЦ ТО**

Тогда для связей 2 и 4 можем применить формулу (2.4) учета окружающей среды рассматриваемой системы (Рисунок 2.3). Здесь связь 4 будет отождествлять

собой среду $S_{T2}^{CS}$ для функционирования разработанной СЛУ в режиме отладки на этапе тестирования на испытательном стенде (Рисунок 2.3):

$$S_{T2}^{CS} =< S_{T1}^{CS}, Z_T^{CS}, R_{T2}^{CS} >, \quad (2.23)$$

где $Z_T^{CS} =< SR_T^{CS}, q(SR_T^{CS}), R_{TZ}^{CS} >$ – это структура, описывающая множество элементов среды $SR_T^{CS}$, включающее готовый испытательный стенд, множество свойств элементов среды $q(SR_T^{CS})$ и множество связей между компонентами этой структуры $R_{TZ}^{CS}$; $R_{T2}^{CS}$ – множество связей между компонентами системы $S_{T2}^{CS}$.

В свою очередь связь 2 будет отображать среду $S_{IPC2}^{CS}$ для функционирования готовой отлаженной СЛУ в качестве системы электроавтоматики в изготовленном технологическом оборудовании (Рисунок 2.3):

$$S_{IPC2}^{CS} =< S_{IPC1}^{CS}, Z_{IPC}^{CS}, R_{IPC2}^{CS} >, \quad (2.24)$$

где $Z_{IPC}^{CS} =< SR_{IPC}^{CS}, q(SR_{IPC}^{CS}), R_{IPCZ}^{CS} >$ – это структура, описывающая множество элементов среды $SR_{IPC}^{CS}$, включающее готовое ТО, множество свойств элементов среды $q(SR_{IPC}^{CS})$ и множество связей между компонентами этой структуры $R_{IPCZ}^{CS}$; $R_{IPC2}^{CS}$ – множество связей между компонентами системы $S_{IPC2}^{CS}$.

Таким образом получаем замкнутый цикл, начало своего функционирования который берет от $S_{DG1}^{EQ}$ – системы объектов и процессов этапа проектирования ТО (Рисунок 2.3). Финальной точкой является $S_{M1}^{EQ}$ – система объектов и процессов этапа производства ТО (Рисунок 2.3).

## 2.4 Построение структурной модели комплекса тестирования систем логического управления

В данной работе предлагается структурная модель комплекса тестирования, включающая в себя три основных компонента: стенд тестирования, аппаратный программируемый логический контроллер и программно реализованный контроллер (Рисунок 2.4).

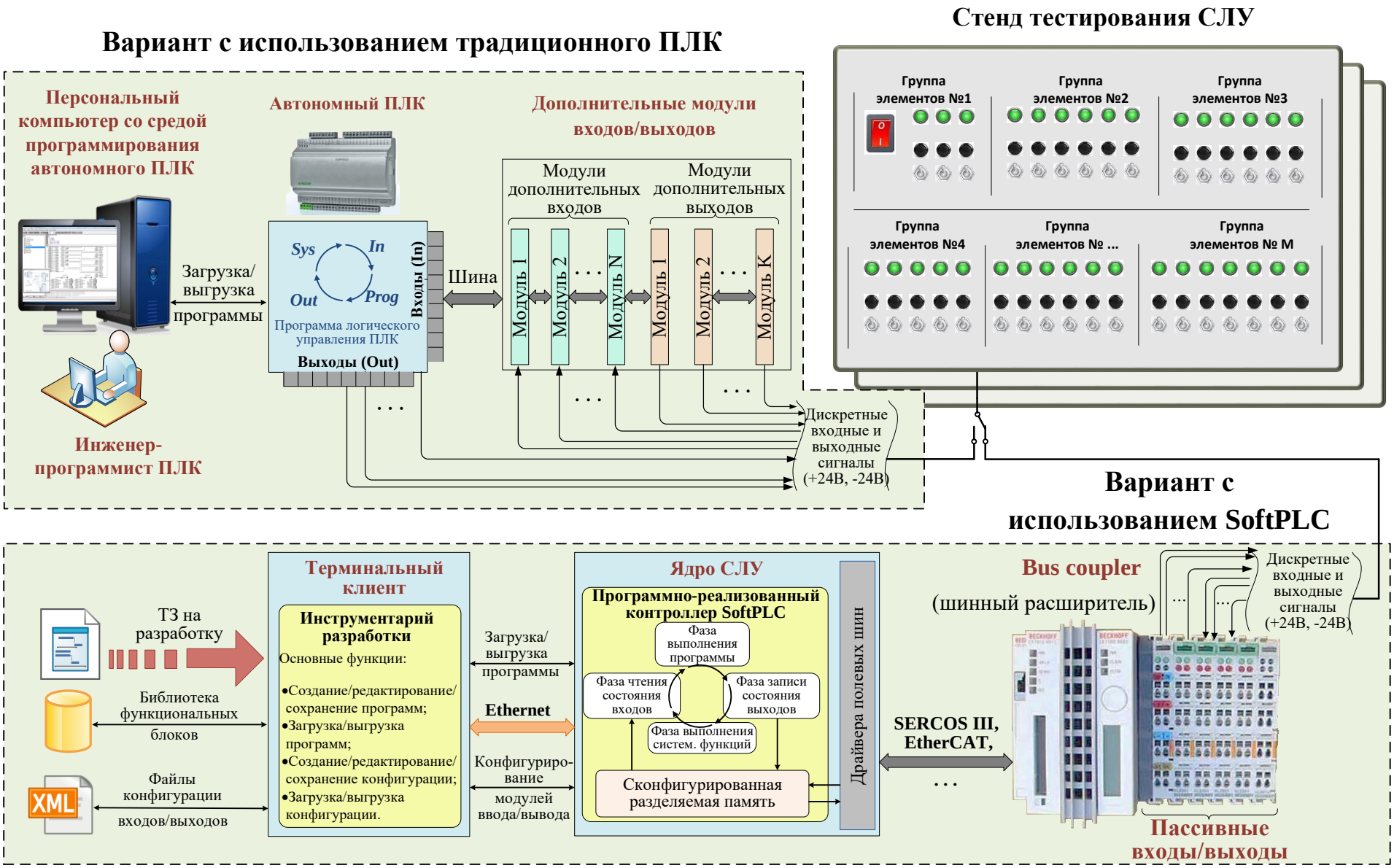



**Рисунок 2.4 – Структурная модель комплекса тестирования систем логического управления**

Поясним наличие данных устройств в модели, а также их состав. Структурная модель комплекса тестирования систем логического управления описывает схему взаимодействия ПЛК со специализированным стендом тестирования систем логического управления. Предпосылки к этому приведены выше в п.п. 2.1 – 2.2.

Описываемая структурная модель ориентирована на возможность использования как традиционных ПЛК (Рисунок 2.4, верхняя часть), так и программно реализованных контроллеров, для отражения особенностей архитектуры наиболее часто проектируемых систем логического управления (Рисунок 2.4, нижняя часть). Такая модель взаимодействия позволит отслеживать ошибки, если таковые имеются, как в исполнительном ядре системы управления (для этого будет использоваться автоматизированное тестирование и предложены тестовые сценарии), так и проводить тестирование программы, реализующей логику работы всей системы логического управления в целом (для этого будет использоваться ручное тестирование и предложены соответствующие тестовые сценарии).

Выбор комбинированного подхода для построения структурной модели обуславливается необходимостью быстрой переналадки комплекса тестирования под максимально возможное количество систем логического управления, требующих тестирования с помощью предлагаемых в работе методики и алгоритмов.

Данная структурная модель состоит из трех частей: непосредственно стенд (стенды) тестирования, соединенный с помощью электрических проводов с дискретными сигналами на входах-выходах либо аппаратного ПЛК, либо программно реализованного. Если СЛУ реализована на базе SoftPLC, то соединение со стендом происходит через блок пассивных входов-выходов, вся логика реализована в ядре СЛУ, а программирование осуществляется через терминального клиента. Если же имеем дело с традиционным ПЛК, программа логического управления загружается с обычного персонального компьютера, оснащенного средой программирования. Сигналы поступают на модуль

дискретных входов ПЛК, программа выполняется в цикле, по итогу ее выполнения формируются выходные сигналы на модуле дискретных выходов ПЛК и далее через электрические связи идут на стенд в виде выходных реакций (вкл/выкл лампочек).

При необходимости, в случае проверки на работоспособность системы управления, имеющей несколько сотен входных-выходных сигналов, с целью недопущения чрезмерного разрастания панели стенда тестирования, возможно использование нескольких стендов тестирования, подключаемых к проверяемой системе поочередно.

Следует также отметить, что, в общем случае, предлагаемая структурная модель, а в дальнейшем и методика, предоставляют возможность обработки как дискретных, так и аналоговых сигналов, поступающих с объекта управления. Но в данной работе круг возможных входных сигналов был ограничен лишь дискретными, так как порядка 75-80% обрабатываемых входных данных на промышленном оборудовании являются цифровыми [1, 64, 42].

Остановимся немного подробнее на двух возможных вариантах использования структурной модели, в зависимости от способа реализации системы управления, и рассмотрим их более детально.

### 2.4.1 Системы логического управления на базе аппаратных программируемых логических контроллеров

Одним из наиболее часто встречающимся вариантом компоновки системы логического управления является использование традиционного аппаратного ПЛК. Именно поэтому было принято решение о включении его в разрабатываемую структурную модель. «Использование автономного программируемого логического контроллера (Рисунок 2.5) целесообразно при разработке систем управления для новых единичных образцов металлорежущего оборудования на основе уже существующего модельного ряда. При этом состав цикловой электроавтоматики, аппаратные средства автоматизации и соответствующая

программная реализация алгоритмов управления остаются практически идентичными, что позволяет минимизировать изменения в проекте электрооборудования разрабатываемого образца» [46].

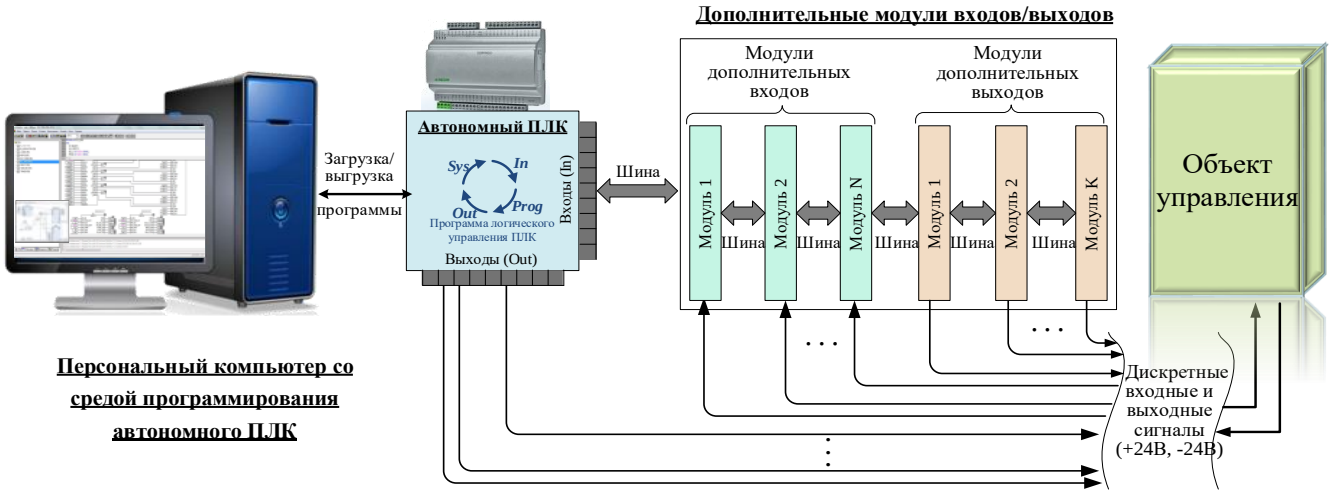



**Рисунок 2.5 – Структура и организация СЛУ на базе традиционного ПЛК**

В общем случае контроллеры могут обрабатывать как цифровые, так и аналоговые сигналы. В данной работе рассматриваются только дискретные входные и выходные сигналы. Они составляют 90% всех сигналов систем управления технологическим оборудованием, а для несложных систем (общее количество входов-выходов – не более 100) – все 100%.

Для случая подключения системы управления, работающей на базе аппаратного ПЛК, к стенду тестирования имеем подключение дискретных входов/выходов стенда непосредственно к входам/выходам ПЛК. Загруженная через подключенный компьютер со средой программирования контроллера программа логического управления начинает свою работу, функционируя в цикле. В качестве входных данных поступают сигналы от тумблерных или кнопочных переключателей стенда на модуль дискретных входов ПЛК. Программа логического управления, используя заданный тестировщиком набор дискретных входных значений, работает в соответствии с заложенным алгоритмом и генерирует, в свою очередь, набор дискретных выходных значений, воспринимаемых стендом через линии связи как сигнал для включения или

выключения соответствующих световых индикаторов (лампочек). Полученные выходные значения сравниваются с требуемыми и на основании этого делается вывод об успешном (или провальном) прохождении тест-кейса. По итогу обработки всех заданных наборов входных значений и фиксации выходных формируется отчет об ошибках, оформляемый в виде таблицы.

### 2.4.2 Системы логического управления на базе программно реализованных логических контроллеров

Контроллер типа SoftPLC (Рисунок 2.6) подходит для построения систем управления гаммы разрабатываемых станков. Вариации в конструктивном исполнении технологического оборудования требуют оперативного изменения программы управления. Использование SoftPLC дает такую возможность. В качестве среды для создания управляющих программ будем и пользовать FBEditor, который позволяет быстро и удобно создавать, редактировать и обновлять управляющие программы, в соответствии с изменяющими условиями и требованиями объекта управления [46, 83, 48].

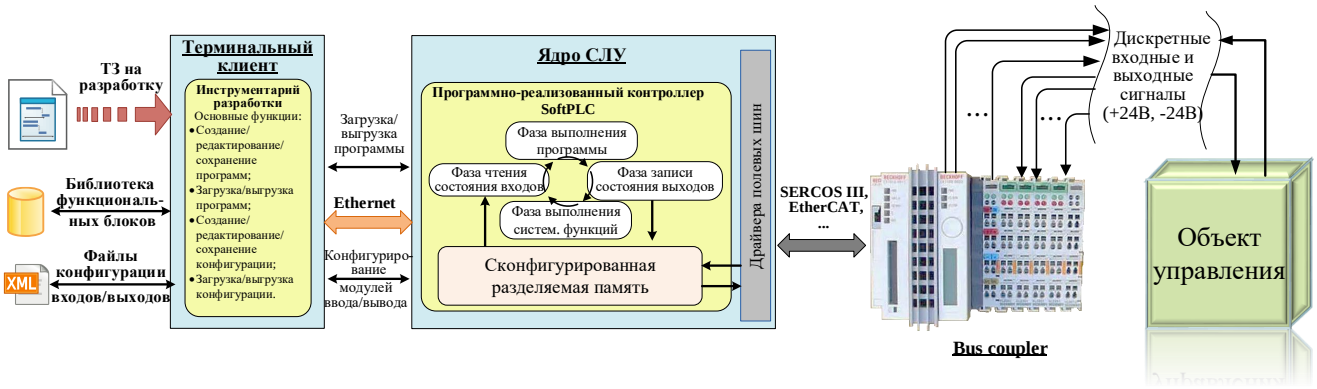



**Рисунок 2.6 – Структура и организация СЛУ на базе SoftPLC**

На основе технического задания с использованием терминального клиента, который включает в себя весь инструментарий среды FBEditor, составляется управляющая программа на языке FBD (Function Block Diagram, один из пяти языков программирования ПЛК, определенных стандартом МЭК 61131-3 [23]). В общем случае она состоит из:

- «основной программы, хранящейся в файле с расширением «*.fbv»» [88],
- «множества библиотек пользовательских подпрограмм, хранящихся в файлах с расширением «*.fbl»» [88];
- файлов конфигурации входов/выходов системы расширением «*.xml».

В ходе работы системы, после проведения ряда системных проверок, программа логического управления объединяется с входящими в нее пользовательскими подпрограммами (в своем изначальном виде), происходит компоновка программы. В подсистему логического управления загружается объединенная управляющая программа и после успешного прохождения еще одного этапа проверок производится запуск программы в цикле работы логического контроллера.

Помимо непосредственно самой управляющей программы, отвечающей за логику работы SoftPLC, в ядро системы управления загружаются файлы конфигурации входов/выходов системы, имеющие расширение «*.xml». Правильная настройка входов/выходов имеет такое же важное значение для правильной работы всей системы, как и написание управляющей программы, отвечающей всем требования технического задания. Корректное выполнение обоих процессов будет залогом адекватной работы системы логического управления в целом. Так как в случае использования программно реализованного контроллера не используются блоки расширения входов/выходов, как при работе с аппаратным ПЛК, то для принятия всех возможных входных сигналов с объекта управления, а также выдачи управляющих воздействий от SoftPLC на исполнительные органы механизмов управления используется блок пассивных входов/выходов (bus coupler), который подсоединяется к ядру СЛУ через драйверы полевых шин с использованием промышленных протоколов связи (например, SERCOS III, EtherCAT и др.).

Суть работы SoftPLC с загруженной в него программой управления (с учетом некоторой специфики, накладываемой виртуальной реализацией ПЛК) также

сводится к циклической обработке входных сигналов в соответствии с управляющей программой и выдаче выходных управляющих воздействий.

Разработки в направлении программно реализованного логического контроллера являются новым направлением в области автоматизации. SoftPLC имеет ряд преимуществ по сравнению с традиционным ПЛК:

- не является дополнительным оборудованием, поэтому техническая поддержка и сопровождение осуществляется совместно с обслуживанием программно-аппаратного обеспечения системы управления в целом;
- представляет собой программно-математическое обеспечение в рамках общей вычислительной системы, а значит, имеет возможность тесного взаимодействия как с другими задачами управления, так и со всеми модулями системы [17];
- возможность быстрой модернизации системы без длительной остановки и наладки аппаратной компоненты СЛУ за счет обновления программного обеспечения;
- при программной реализации появляется возможность диагностики, установки обновлений и устранения ошибочных ситуаций посредством удаленной работы через Internet [74, 81].

Одним из вариантов реализации SoftPLC является программно реализованный промышленный контроллер, разработанный на кафедре «Компьютерные системы управления» в ФГБОУ ВО «МГТУ «СТАНКИН». Структура контроллера включает следующие компоненты (Рисунок 2.7):

- среду разработки, где на основе технического задания формируется управляющая программа (УП) и конфигурация оборудования;
- ядро, где происходят вычислительные преобразования модели;
- драйверы внешних устройств, с помощью которых командные сигналы передаются непосредственно на физические устройства (исполнительные механизмы объекта управления) [46].

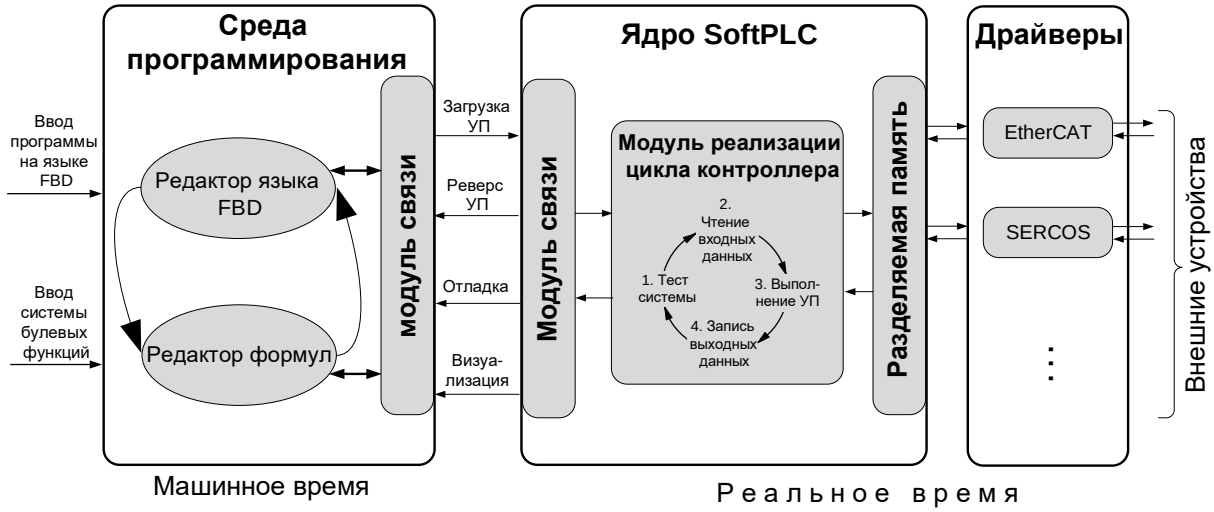



**Рисунок 2.7 – Архитектура программно реализованного логического контроллера**

В состав прикладного программного обеспечения терминальной части входит:

- среда разработки и отладки программ логического управления FBEditor, основным языком программирования которой является язык FBD – Function Block Diagram (входит в стандарт МЭК 61131-3) [23];
- среда конфигурирования оборудования, предназначенная для настройки основных параметров доступа к аппаратным средствам с помощью драйверов внешних устройств и одного из стандартных протоколов связи (например, SERCOS или EtherCAT).

В настоящее время в ФГБОУ ВО «МГТУ «СТАНКИН» продолжаются работы по дальнейшей разработке программно реализованного контроллера.

Схема взаимодействия системы управления, построенной на базе SoftPLC, со стендом тестирования, аналогична описанной в п. 2.4.1. Основное отличие здесь состоит в том, что дискретные входы и выходы стенда соединяется с контроллером не напрямую, а через блок пассивных входов/выходов (bus coupler), о котором упоминали выше. Еще одной особенностью системы управления, использующей программно реализованный контроллер, является то, что все взаимодействие человека с системой и контроллером происходит через

терминального клиента. С помощью него загружается (или выгружается) программа логического управления в контроллер, подгружаются системные и пользовательские библиотеки программ, производится конфигурирование входов\выходов с помощью специальных файлов конфигурации. На этом отличия исчерпываются. Вся остальная схема работы абсолютно идентична тому, как это было описано для аппаратного ПЛК. Основная необходимость использования программно реализованных контроллеров возникает, как было упомянуто выше, когда предполагается работа с широким спектром различных систем управления, что, в свою очередь, потребует частой перенастройки управляющей программы. SoftPLC позволяет это сделать быстрее и удобнее, нежели традиционные ПЛК.

## 2.5 Преимущества применения стендового тестирования

Для введения в эксплуатацию готовой системы необходимо убедиться в корректности работы аппаратного обеспечения и программной реализации алгоритмов управления. Отлаживать готовую систему в производственных условиях сложнее и дольше по времени, чем в условиях офиса. Помимо этого, первоначально программа может содержать ошибки, которые могут привести к поломке дорогостоящих узлов при отладке на реальном технологическом оборудовании. В связи с этим возникает необходимость в разработке методики на базе построения стендов тестирования и отладки аппаратного и программного обеспечения, ориентированных на конкретное архитектурное решение систем логического управления. При этом стенды должны быть недорогими, быстросборными, иметь структурные признаки готовой системы управления (вычислительные ресурсы, количество входов/выходов, имитация узлов технологического оборудования и т.д.). Все эти факторы требуют высокого уровня ответственности и надежности работы систем, а также диктуют необходимость возможности быстрой переналадки и внесения индивидуальных корректировок в

соответствии с изменяющимся функционалом систем управления. Эти и другие факторы определяют следующие преимущества стендового тестирования:

- экономия энергии: комплектация специализированного испытательного стенда, все элементы, участвующие в работе тестирования на нем, являются значительно менее энергозатратными, чем аналогичные процессы, запущенные для проверки на реальном оборудовании;
- снижение влияние человеческого фактора на ряд процессов: данное преимущество наиболее очевидно при использовании автоматизированного тестирования, хотя и в случае ручного тестирования также имеет место быть, но в значительно меньшей степени;
- улучшение условий труда: в основном, за счет автоматизации, большей степени контроля процесса, а также возможности проведения тестов не в условиях постоянного присутствия в цехе (температурный режим ниже комфортного для человека, наличие высокого уровня шума и продуктов работы технологического оборудования), а на отдельном рабочем месте, оснащенном в классическом офисном стиле;
- запуск алгоритмов управления и тестирования без необходимости иметь навык низкоуровневого программирования;
- наглядность анализируемых процессов;
- быстрое, автоматизированное тестирование базовой функциональности системы логического управления;
- индивидуальный подход к каждому объекту управления;
- быстросборность, т.е. быстрое переконфигурирование стенда для других проектов (гибкость в работе, за счет чего достигается скорость исполнения);
- возможность проведения процесса тестирования, в случае если алгоритмы написаны, но использование реального объекта для испытаний *невозможно* (прототип еще не готов, проводить натурные испытания крайне дорого или существует опасность для здоровья тестировщика);

- возможность проведения процесса тестирования, в случае если алгоритмы написаны, но использование реального объекта для испытаний *нецелесообразно* (алгоритмы еще не проверены и могут привести к поломке/разрушению объекта);
- широкий спектр применения одной и той же технологии и одного и того же стенда для разнообразных целей (симуляторы различных условий эксплуатации (аварии, критические режимы), тренажеры для обучения эксплуатирующего персонала);
- стенды хорошо себя зарекомендовали как наглядное учебное пособие в специальных и высших учебных заведениях; такой способ их использования в качестве «второй жизни» после окончания использования на производстве для тестирования СЛУ находит широкое применение;
- возможность проведения испытаний и сокращения издержек на испытания в случае, если:
  - реальный объект недоступен или в единственном экземпляре;
  - далеко ездить на испытания.
- полноценное систематическое тестирование системы (воспроизводимые тесты, тесты на функциональную безопасность и отказоустойчивость).

Таким образом, достоинства применения испытательных стендов очевидны. Их использование позволяет осуществлять комплексную стыковку объектов испытываемой системы и проверку принципов управления задолго до создания всех элементов системы (элемент системы, разработка которого не завершена, заменяется моделью).

Проектирование специализированных испытательных стендов означает сборку и структурирование стенда под конкретный тестируемый объект. Как и любой реально существующий процесс или устройство использование стендов тестирования дает большое количество преимуществ, но имеет и свои ограничения. Перечислим основные из них.

Ограничения по применению стендов при тестировании работы системы управления:

- «Скоростные» входные переключения, т.е. проверка входных сигналов, которые должны подаваться системой последовательно с коротким интервалом, менее 1 секунды (входные сигналы на стенде имитируются переключателями в виде кнопок или тумблеров, переключение которых вручную с интервалом менее 1 с не представляется возможным);
- «Скоростные» выходные переключения, т.е. проверка выходных сигналов, реакция системы на которые не превышает 1 с, а также ряд последовательных выходных срабатываний, которые должны среагировать друг за другом с коротким интервалом, менее 1 секунды (возможности зрительного восприятия не позволяют с таким коротким интервалом времени визуально отследить быстрые переключения лампочек, реализующих срабатывание тех или иных выходных управляющих сигналов).

Стоить уточнить, что предлагаемые в работе модель и методика применимы и для систем управления, включающих в себя обработку аналоговых сигналов. В таком случае для проектирования стенда, кроме световых индикаторов и реле-переключателей, понадобятся дополнительные элементы, позволяющие имитировать и детектировать аналоговые сигналы. Однако в данной работе предлагаемые разработанные стенды охватывают только дискретные сигналы, так как в случае промышленного оборудования таких большинство.

Отладка программных продуктов с помощью специализированных испытательных стендов дает большое количество достоинств, но имеет и свои недостатки, которые нужно учитывать в случае выбора их для проверки работоспособности каждого определенного проекта. Описанные выше ограничения решаются с помощью использования дополнительных устройств, позволяющих детектировать подобные быстрые срабатывания. Однако применение такого рода устройств в концепции предлагаемого решения нецелесообразно с экономической и технической точки зрения. Использование стенда тестирования позволяет отработать до 90% сигналов, а для некоторых

относительно несложных станков и все 100%. Именно поэтому данному решению уделяется достаточное внимание в предлагаемой работе.

## 2.6 Выводы к главе 2

1. Стендовое тестирование является удобным и перспективным способом безопасной проверки работоспособности систем логического управления, имеющим как ряд преимуществ, так и некоторые ограничения.

2. Программируемый логический контроллер является наиболее широко используемым инструментом для разработки СЛУ и формирования логики их работы.

3. Комплекс мероприятий по разработке нового технологического оборудования можно представить в виде системы жизненных циклов трех объектов: готовое ТО с установленной СЛУ, СЛУ, стенд тестирования для проверки работоспособности СЛУ. Применение системного подхода к рассматриваемым циклам позволило разработать унифицированное формальное описание процессов и связей ЖЦ ТО на трех уровнях детализации.

4. Из всего комплекса процедур по разработке и созданию технологического оборудования для изучения в рамках работы к детальному рассмотрению предлагаются этапы создания программы логического управления, тестирования программы ЛУ системы с помощью разработанного и собранного специализированного стенда, а также последующая доработка программного обеспечения и конфигурация входов/выходов на основе результатов тестирования.

5. Формализованное представление наглядно отражает взаимосвязи ЖЦ компонент ТО, которые являются основой для разработки структурной модели комплекса тестирования СЛУ с использованием специализированного стенда тестирования, ориентированной на возможность использования как традиционных ПЛК, так и программно реализованных контроллеров.

6. Включение в структурную модель специализированного стенда тестирования дает целый спектр преимуществ как на этапе разработки системы логического управления, так и на этапе ее непосредственного тестирования.

# ГЛАВА 3. РАЗРАБОТКА МЕТОДИКИ, АЛГОРИТМОВ И СЦЕНАРИЕВ СТЕНДОВОГО ТЕСТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ

Как уже было отмечено во второй главе, предлагаемая структурная модель комплекса тестирования систем логического управления позволяет тестировать как СЛУ на базе автономного программируемого логического контроллера, так и системы автоматизации, построенные на основе программно реализованного SoftPLC. При этом предлагаемая структурная модель, и в том, и в другом случае, подразумевает применение стендового тестирования. Выполнение процесса проверки на работоспособность предлагается с помощью разработанных методики и алгоритмов.

## 3.1 Разработка методики тестирования систем логического управления с использованием специализированных испытательных стендов

Реализация предлагаемого подхода предполагает разработку методики, включающей в себя несколько последовательных шагов, изображенных на рисунке 3.1. Данные шаги являются результатом подробного изучения различных источников по практическим действиям при разработке новых программных продуктов и систем либо их модернизации. Ниже представлено более подробное описание каждого из этапов.

Приступая к реализации данной методики за исходные данные берутся: техническое задание на разработку, программа логического управления тестируемой СЛУ, а также предварительная (ориентировочная) конфигурация входов/выходов контроллера (аппаратного или SoftPLC), на основе которых производится анализ автоматизированной системы с целью отнесения ее к одному из четырех возможных вариантов (шаг 1). Чтобы это сделать, поясним, что одним из основных признаков классификации СЛУ является требуемое количество

входов/выходов, необходимое для корректной реализации логики работы системы автоматизации в соответствии с техническим заданием.

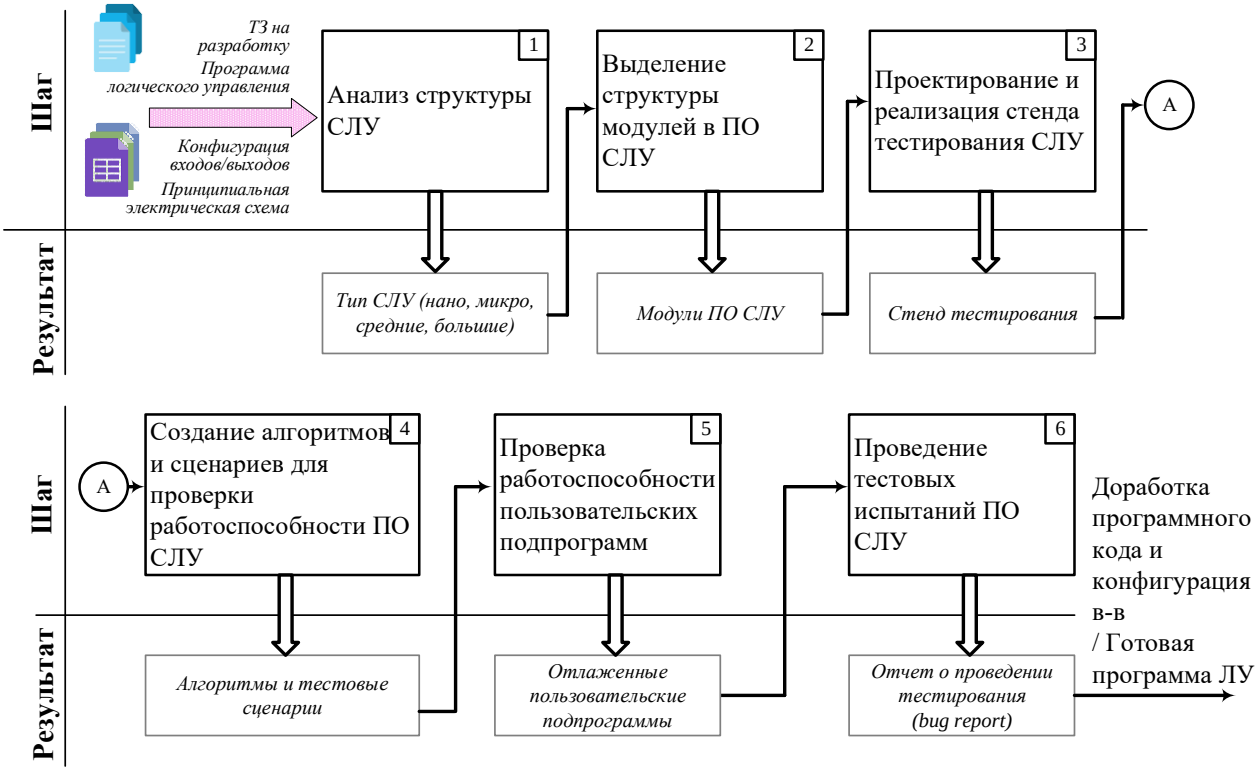



**Рисунок 3.1 – Методика стендового тестирования систем логического управления**

Работая на первом шаге методики следует знать, что по данному критерию все системы логического управления можно разделить на 4 основные группы:

- *нано системы (nano)* – менее 15 точек ввода/вывода;
- *микро системы (micro)* – 15-128 точек ввода/вывода;
- *средние системы (medium)* – 128-512 точек ввода/вывода;
- *большие системы (big)* – более 512 точек ввода/вывода.

Таким образом, результатом первого шага будет принятое решение о том, с системой какого типа предстоит работать на дальнейших этапах реализации методики тестирования.

Различными научными сообществами проводились исследования с целью понять, какие из этих четырех типов СЛУ являются наиболее востребованными на

рынке автоматизированных систем. На Рисунке приведены результаты исследований, которые показали наиболее часто используемые варианты СЛУ (Рисунок 3.2).

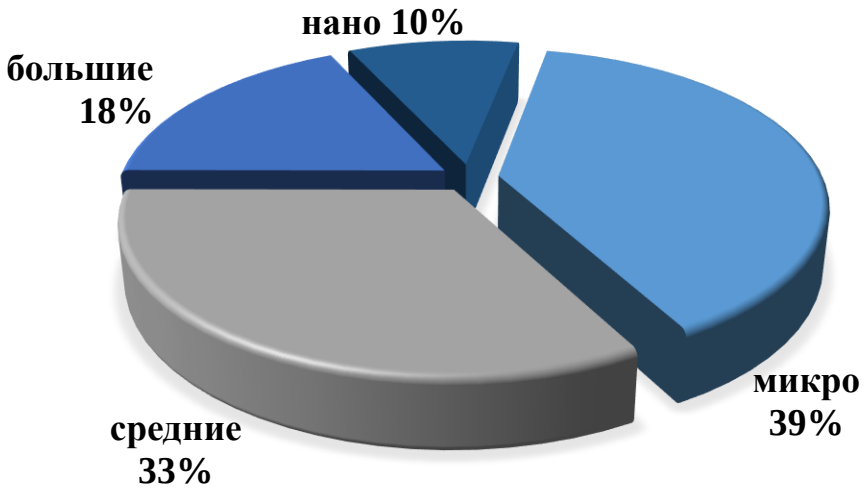



**Рисунок 3.2 – Диаграмма распределения различных видов СЛУ по популярности использования**

Из рисунка ясно видно, что чаще всего востребованы микро и средние системы. Данный факт будет полезен при осуществлении второго шага методики.

Диаграмма также показывает, что наиболее часто встречаемые варианты систем управления предполагают наличие от 15 до 512 входов/выходов. Поэтому следующим шагом методики будет выделение структуры модулей в программном обеспечении СЛУ (шаг 2). Причем в рамках данной работы будем считать, что для СЛУ, имеющей не более 100 входов/выходов тестирование проводится на стенде, полностью повторяющем все входы/выходы реальной СЛУ. При наличии в испытываемой системе более 100 входов/выходов выполнять проверку на стенде, полностью имитирующем реальную СЛУ, не представляется целесообразным, тестирование таких систем будет проводиться с выделением отдельных логических модулей с использованием модульного тестированиям (Рисунок 3.3). Выбор производится в зависимости от конкретно поставленной решаемой задачи.

Далее на основе сформированных модулей системы логического управления, с учетом их группировки по назначению либо по основным функциональным узлам объекта управления, разрабатывается и собирается стенд тестирования (шаг 3).


**Рисунок 3.3 – Разбиение всех СЛУ на две категории для реализации стендового тестирования**

Структурно стенд включает в себя те же группы элементов, которые были получены при разделении системы на логически обособленные компоненты (модули). Более подробно разработка стенда будет рассмотрена в главе 4.

Наличие готового стенда тестирования позволяет перейти к следующему шагу методики, представляющему собой разработку тестовых сценариев для проверки работоспособности программы логического управления СЛУ (шаг 4). Для реализации стендового тестирования будем использовать методы проверки, сформулированные в качестве решения в главе 1 (Таблица 1.5). Алгоритмы и сценарии тестирования подробно будут описаны в пп. 0.

Пятый этап методики подразумевает проверку работоспособности пользовательских библиотек, являющихся частью основной тестируемой программы логического управления. Его необходимость обоснована тем, что в процессе разработки данного и других ранее создаваемых программных проектов формируются некоторые шаблоны, библиотеки и пользовательские подпрограммы, используемые многократно в различных системах. Прежде чем приступить к тестированию прикладного решения текущего объекта управления, необходимо проверить, что все используемые библиотеки работают корректно и соответствуют

текущей операционной и текущей версии среды разработки управляющих программ. Успешное завершение 5 шага позволит приступить к заключительному, 6му – тестированию непосредственно программы логического управления в целом на специализированном разработанном стенде с помощью составленных тестовых сценариев и набора тест-кейсов.

Под *тестовым сценарием (тест-план)* здесь будем понимать четко сформулированный, заранее спланированный формализованный подход к проверке работоспособности программы логического управления с помощью заранее подготовленных тест-кейсов, базирующийся на технической документации проекта. Тестовый сценарий определяет, в какой последовательности будут запускаться тесты, выбранные для комплексного тестирования данной СЛУ, как изменится последовательность в случае появления тех или иных промежуточных результатов тестирования, как следует фиксировать отчеты об ошибках, каким образом интерпретировать выявление тех или иных дефектов, нерегулярных ситуаций, какие действия предпринимать после формирования отчета о проведенном тестировании, а также что необходимо для выполнения шагов тесто-плана. Выполняется, как правило, в виде таблиц, содержащих входные и соответствующие им выходные значения. Для удобства последующего использования каждый тестовый сценарий должен обладать названием (должно быть кратким и понятным, описывающим суть проверки).

В свою очередь *тест-кейс (тестовый случай)* – это непосредственно набор данных (один из многих) для проведения того или иного метода проверки (тестовая, моделируемая ситуация). Для удобства последующего использования и быстрой навигации по таблицам (тест-наборам) каждый тест-кейс должен обладать номером (уникальным идентификатором тест-кейса), последовательностью шагов, необходимых к выполнению для реализации данного тестового случая, и ожидаемый результат (что мы ожидаем увидеть после выполнения шагов).

Результатом выполнения 6 шага методики тестирования систем логического управления является сформированный отчет об ошибках (bug report) (Таблица 3.1, Рисунок 3.5). *Отчет об ошибках* – это технический документ, описывающий

ситуацию и/или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата. «В задачи, решаемые в рамках составления отчета об ошибках, входят:

- предоставление информации о проблеме, ее свойствах и последствиях;
- приоритизация проблемы по важности и скорости устранения;
- помощь программистам в обнаружении и устранении источника проблемы» [79].

**Таблица 3.1 – Пример содержания отчета о результатах тестирования**

| **Компоненты отчета** | **Содержание компонента отчета** |
|---|---|
| *1* | *2* |
| Краткое описание проблемы | Короткое описание проблемы, явно указывающее на причину и тип ошибочной ситуации |
| Проект | Название тестируемого проекта |
| Компонент приложения | Название части или функции тестируемого продукта |
| Серьезность | Наиболее распространена пятиуровневая система градации серьезности дефекта:<br>1. блокирующий<br>2. критический<br>3. значительный<br>4. незначительный<br>5. низкий |
| Приоритет | Приоритет дефекта:<br>1. высокий<br>2. средний<br>3. низкий |
| Статус | Статус ошибки; определяет текущее состояние дефекта. Зависит от используемой процедуры и выбранной системы отслеживании ошибок (bug-tracking system). |

«Отчет об ошибке («баг-репорт», «bug report», баг-трекинг) – один из основных результатов работы тестировщиков. Именно этот результат работы видят коллеги (другие тестировщики и люди, не входящие в команду

тестировщиков)» [79]. Но не стоит забывать, что основная цель написания отчета об ошибках – устранение ошибок. Поэтому написание за день 1000 бесполезных и бессмысленных отчетов не будет показателем профессионализма специалиста по тестированию. Главным критерием показателя результативности его работы будет то, сколько ошибок было исправлено по его отчетам.

По итогам проведения 5го шага методики тестирования СЛУ на основе отчета об ошибках принимается решение о необходимых мерах по устранению обнаруженных ошибок, уязвимостей и нерегулярных ситуаций.

Придерживаясь данной структуры при составлении отчета о проведенном тестировании, в результате получают отчеты вида как на Рисунках 3.4 или 3.5.

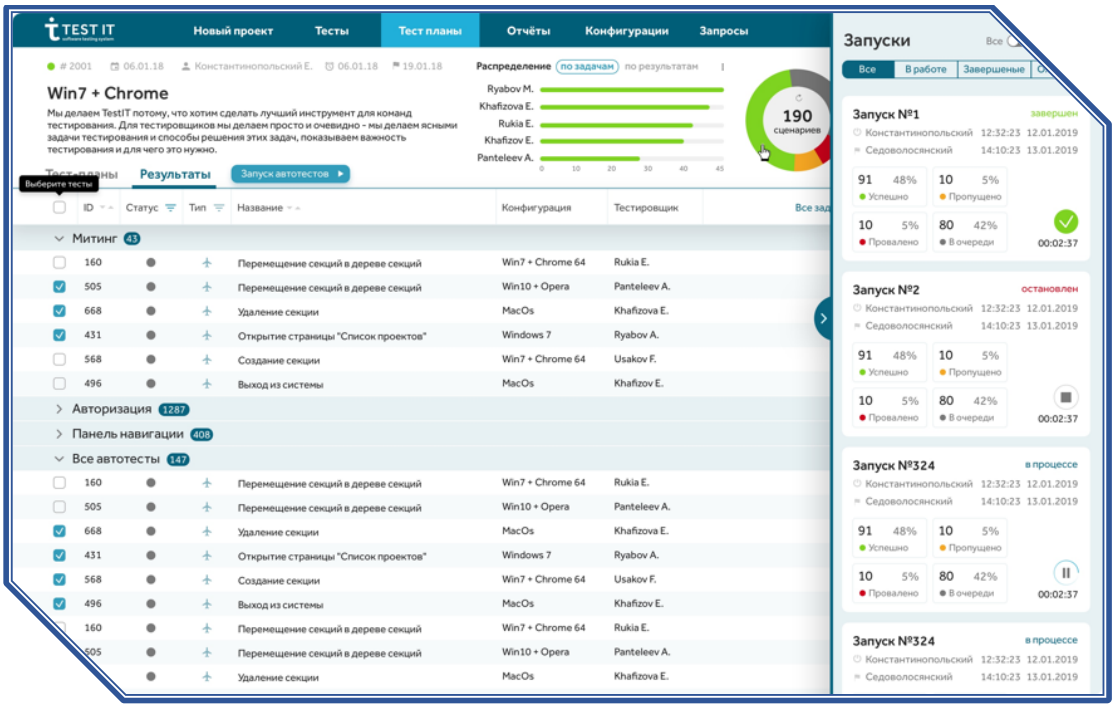
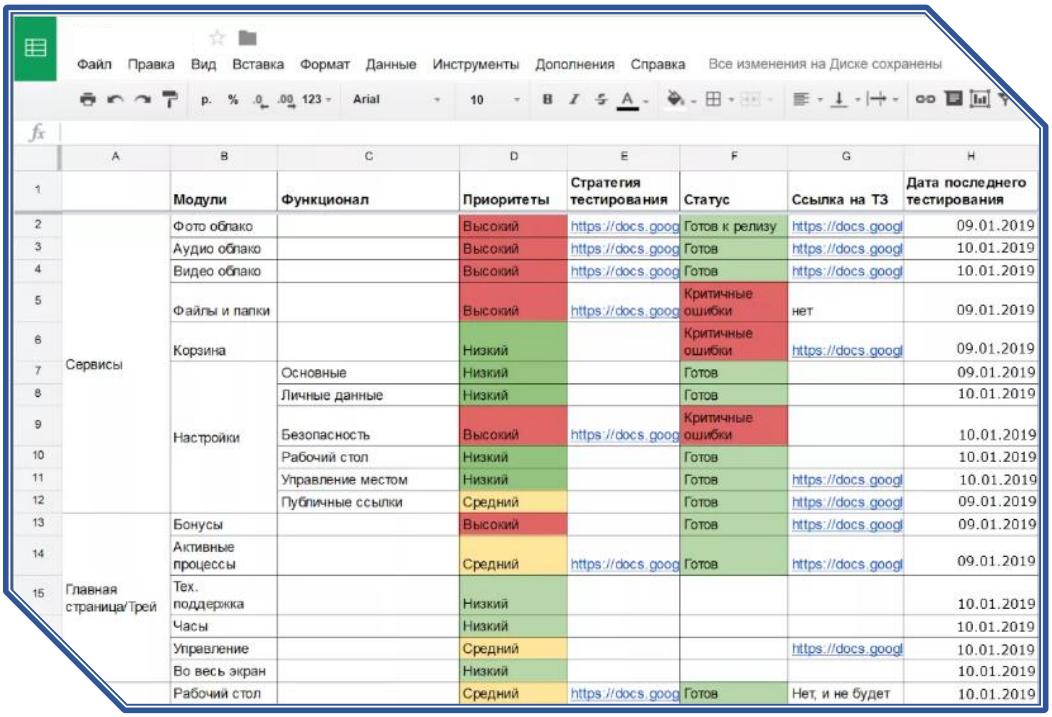



**Рисунок 3.4 – Пример сформированного отчета об ошибках в системе управления тестированием Test IT (фрагмент)**

Для формирования подобных отчетов об ошибках могут использоваться как специализированные программные среды или системы управления тестированием (ALM Octane, Test IT, TestRail, Allure, MTM TFS и др.), так и простые редакторы таблиц.



Файл Правка Вид Вставка Формат Данные Инструменты Дополнения Справка Все изменения на Диске сохранены

| | | Модули | Функционал | Приоритеты | Стратегия тестирования | Статус | Ссылка на ТЗ | Дата последнего тестирования |
|---|---|---|---|---|---|---|---|---|
| 2 | Сервисы | Фото облако | | Высокий | https://docs.goog | Готов к релизу | https://docs.googl | 09.01.2019 |
| 3 | | Аудио облако | | Высокий | https://docs.goog | Готов | https://docs.googl | 10.01.2019 |
| 4 | | Видео облако | | Высокий | https://docs.goog | Готов | https://docs.googl | 10.01.2019 |
| 5 | | Файлы и папки | | Высокий | https://docs.goog | Критичные ошибки | нет | 09.01.2019 |
| 6 | | Корзина | | Низкий | | Критичные ошибки | https://docs.googl | 09.01.2019 |
| 7 | | Настройки | Основные | Низкий | | Готов | | 09.01.2019 |
| 8 | | | Личные данные | Низкий | | Готов | | 10.01.2019 |
| 9 | | | Безопасность | Высокий | https://docs.goog | Критичные ошибки | | 10.01.2019 |
| 10 | | | Рабочий стол | Низкий | | Готов | | 10.01.2019 |
| 11 | | | Управление местом | Низкий | | Готов | https://docs.googl | 10.01.2019 |
| 12 | | | Публичные ссылки | Средний | | Готов | https://docs.googl | 09.01.2019 |
| 13 | Главная страница/Трей | Бонусы | | Высокий | | Готов | https://docs.googl | 09.01.2019 |
| 14 | | Активные процессы | | Средний | https://docs.goog | Готов | https://docs.googl | 09.01.2019 |
| 15 | | Тех. поддержка | | Низкий | | | | 10.01.2019 |
| | | Часы | | Низкий | | | | 10.01.2019 |
| | | Управление | | Средний | | | https://docs.googl | 10.01.2019 |
| | | Во весь экран | | Низкий | | | | 10.01.2019 |
| | | Рабочий стол | | Средний | https://docs.goog | Готов | Нет, и не будет | 10.01.2019 |

**Рисунок 3.5 – Пример сформированного отчета об ошибках в редакторе**

## 3.2 Создание алгоритмов и сценариев тестирования систем логического управления

Разработка методики тестирования систем логического управления с использованием специализированных испытательных стендов

Создание алгоритмов тестирования рассмотрим на примере процедуры проверки функциональной работоспособности исполнительного ядра системы логического управления на базе SoftPLC, изображенную на Рисунке 2.6. Как было сказано выше, одним из вариантов реализации SoftPLC является программно реализованный промышленный контроллер, разработанный на кафедре «Компьютерные системы управления» в ФГБОУ ВО «МГТУ «СТАНКИН». В настоящее время продолжается его разработка, поэтому возникает необходимость многократного проведения тестовых испытаний как программной составляющей, так и системы логического управления в целом. Являясь, по существу, программным обеспечением, оно находится в процессе постоянной доработки существующего функционала и добавления новых функций. Тестирование

управляющей программы для контроллера будем проводить в среде FBEditor, составленной на языке программирования FBD.

### 3.2.1 Автоматизированное функциональное тестирование

Программа на языке FBD представляет собой совокупность функциональных блоков (functional blocks, FBs), которые соединяются линиями связи (connections). Каждый блок представляет собой математическую или логическую операцию. Начальные значения переменных задаются с помощью специальных блоков – входов или констант, выходные цепи могут быть связаны либо с физическими выходами контроллера, либо с глобальными переменными программы.

Для осуществления процесса тестирования составляется управляющая программа на языке FBD, которая, в общем случае, состоит из основной программы, хранящейся в файле с расширением «*.fbv», и множества библиотек пользовательских подпрограмм, хранящихся в файлах с расширением «*.fbl». В ходе работы системы происходит компоновка программы (программа логического управления объединяется с входящими в нее пользовательскими подпрограммами). В подсистему логического управления загружается уже объединенная управляющая программа.

В качестве примера реализации *автоматизированного* функционального тестирования используем логические блоки среды FBEditor и составленный из них пользовательский блок, реализующий логическую функцию Импликации (A->B). Учитывая бинарный характер входов и выходов логических операций, а значит и не слишком большой набор возможных сочетаний входов-выходов блока (тест-кейсов), имеем возможность автоматизировать процесс тестирования логических блоков программно реализованного логического контроллера. Исчерпывающий набор необходимых тест-кейсов как бы инкапсулируется внутрь пользовательского автоматизированного блока тестирования.

Для этого в среде программирования однократно составляется управляющая программа из логических блоков, объединенная в единый пользовательский блок, которому назначаются определенные входы и выходы.

Суть данного алгоритма заключается в следующем. Для инициализации процесса тестирования необходимо однократно подать на вход блока два управляющих сигнала. Эти сигналы, последовательно проходя через базовые тестируемые логические блоки среды программирования, а также через блоки сравнения 1, 2 и 3 уровней, формируют на выходе сигнал в виде логического 0 или 1. Успешное тестирование фиксируется при достижении логической 1 на выходе пользовательского блока. Если же значение на выходе равно 0 – тестовый сценарий считается непройденным, блок работает с ошибкой. На рисунке 3.6 представлена общая схема алгоритма работы автоматизированного тестирования функциональных блоков [15].

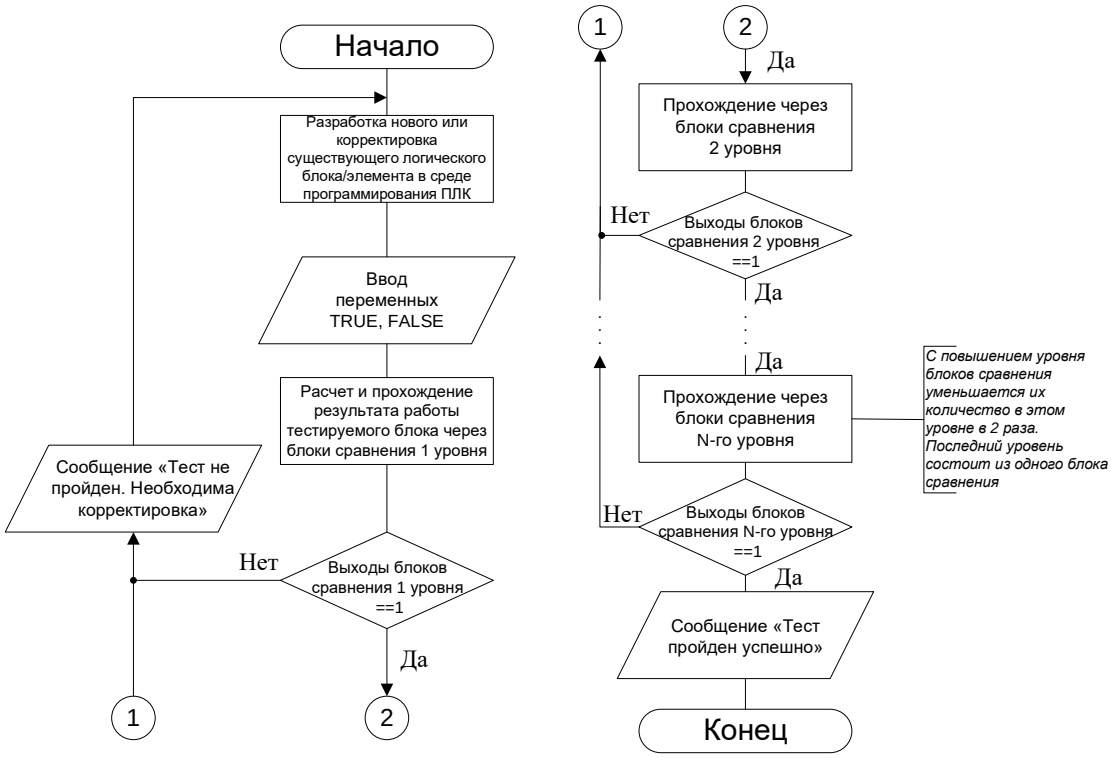



**Рисунок 3.6 – Алгоритм работы автоматизированного тестирования функциональных блоков**

Реализация процедуры автоматизированного тестирования проиллюстрирована на примере проверки корректности работы одного из пользовательских программных шаблонов «Тест Impln-блок» в среде FBEditor (Рисунок 3.7). Также на рисунке представлена внутренняя структура пользовательского блока «Тест Impln-блок».

Данный блок реализует работу булевой функции «Импликация». Ее реализация осуществляется за счет набора однотипных блоков, в данном случае, блоков И, блоков НЕ и блоков Сравнения. Путем постепенного объединения выходов данных блоков получаем единый булевы выход, значение которого может принимать два значения – 1 или 0. В случае автоматизированного тестирования достаточно на имеющиеся два входа однократно подать два сигнала – 0 и 1. Далее все возможные комбинации этих сигналов производятся автоматически внутри пользовательского блока, формируется выходной сигнал. 1 на выходе говорит об успешном тестировании, 0 – ошибка.

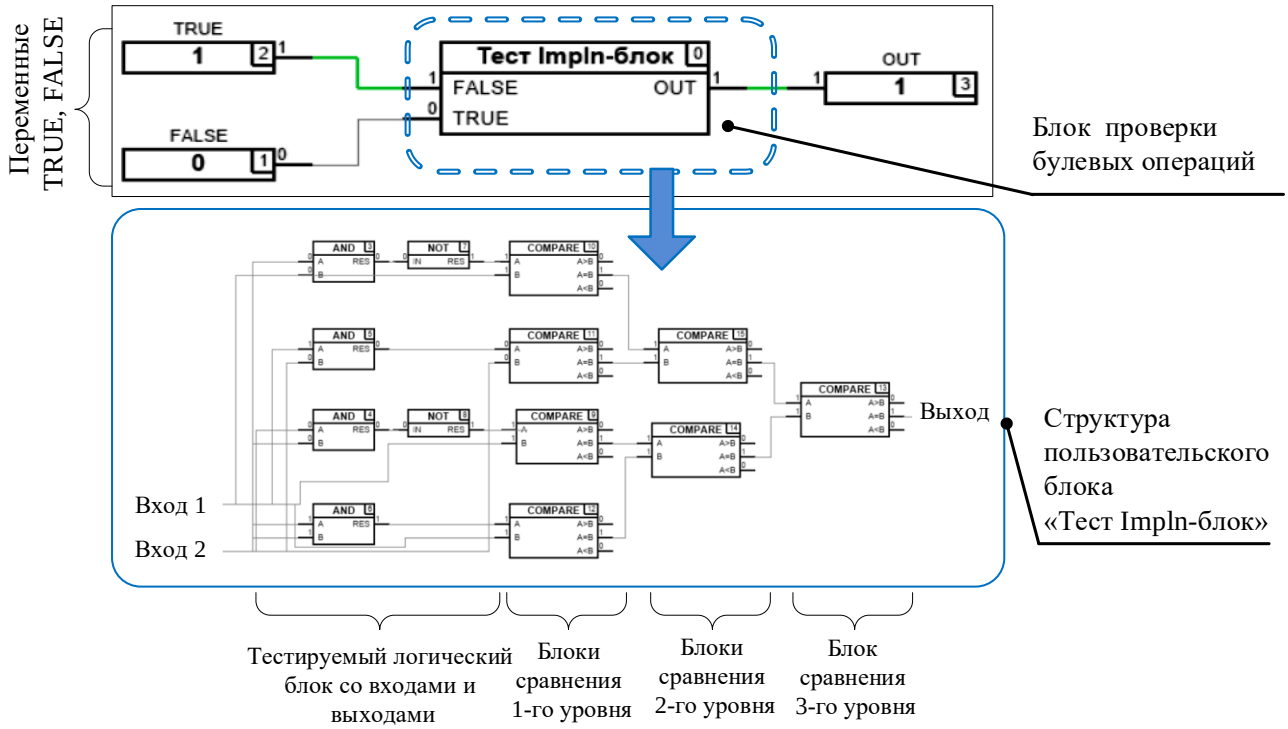



**Рисунок 3.7 – Автоматизированное тестирование функциональных блоков**

Важно отметить, что автоматизированное тестирование предполагает минимальное участие оператора в процессе тестирования. Это позволяет уйти от рутинной работы, исключить «человеческие» ошибки, связанные, например, с неправильной интерпретацией действия той или иной команды, а также существенно сокращает временные затраты. При этом не требуется абсолютно никаких знаний внутреннего содержания тест-блока, нет необходимости в высокой квалификации оператора или владения какими-либо специальными знаниями.

Справедливости ради стоит отметить основной недостаток автоматизированного тестирования. Вернее сказать, не самого процесса проверки, а подготовки его к запуску. Если кратко, этот минус можно назвать «копипаст нон-стоп», то есть комбинация «копировать-вставить» какую-либо область, текст, числа и пр., встречающаяся огромное количество раз. Это необходимо для:

- заполнения приветственных фраз на входе системе (тестируются различные варианты вводимых верных и неверных входных данных);
- при внесении каких-либо изменений в интерфейсе тестируемой системы (тесты кейсы потребуется актуализировать под новый вариант названий вкладок, кнопок и т.п.);
- систем, спроектированных для решения однотипных задач (в интерфейсе будет присутствовать много схожих системных сообщений, высвечивающихся при разнообразных последовательностях возникающих ситуаций).

### 3.2.2 Функциональное тестирование в ручном режиме

Для примера реализации ручного тестирования рассмотрим тестирование блоков арифметических операций среды FBEditor, а именно, сложение, вычитание, умножение и деление. Данные операции предполагают большое количество возможных тестовых случаев, а также число граничных значений и классов эквивалентности (положительные, отрицательные числа, ноль и их комбинации), в отличие логических операций. Поэтому для арифметических операций наиболее

подходящим является *ручное функциональное тестирование*. Проверку работоспособности блоков сложения, вычитания, умножения и деления в разрабатываемой среде для программирования виртуального контроллера выполним в едином блоке путем различных комбинаций этих четырех операций, получив таким образом пользовательский блок для расчета значения среднего арифметического. На Рисунке 3.8 представлена общая схема алгоритма работы ручного тестирования арифметических функциональных блоков [15].

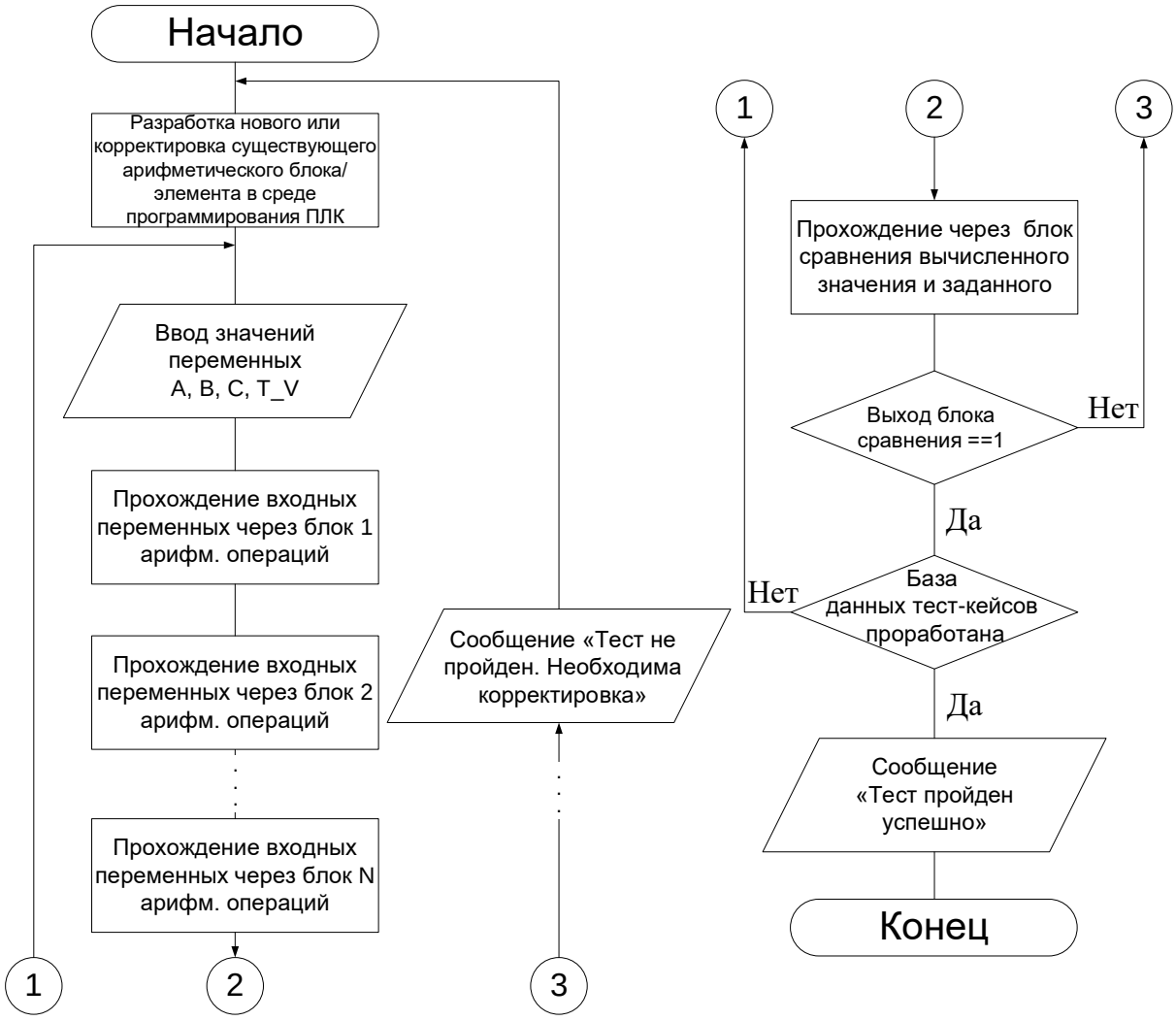



**Рисунок 3.8 – Алгоритм работы ручного тестирования функциональных блоков**

Для выполнения всестороннего тестирования формируется набор тестовых сценариев, которые последовательно отрабатываются оператором.

На входы A, B, C, T_V (первые три из них – исходные числа для расчета среднего арифметического, последнее – целевое значение, т.е. то число, с которым сравнивается полученное расчетное) блока тестирования (Рисунок 3.8) подаются значения из заранее подготовленных тест-кейсов, которые обрабатываются в соответствии с заданным алгоритмом. Затем выходное значение сравнивается с заранее рассчитанным, зафиксированным также в таблице тест-кейса (вход T_V). По пути следования до конечного выходного блока начальные числа проходят через четыре арифметических блока, соединенных по определенному алгоритму, а также через блок сравнения. Заключительный блок сравнения выводит результат своей работы на выходной блок. Если значения двух входов данного блока сравнения совпадают (а это есть расчетное и заданное целевое значения), то на выходе управляющей программы фиксируется 1 – тестирование пройдено успешно. В противном случае – если на выходе получается 0 – тест не пройден, в работе блока фиксируется наличие ошибки.

На Рисунке 3.9 проиллюстрирован пример реализации алгоритма работы ручного тестирования функциональных блоков в среде FBEditor применительно к проверке работы базовых арифметических операций за счет формирования соответствующего пользовательского блока.

В таблице 3.2 приведен фрагмент тестового сценария для него. Значение T_V, после прохождения через несколько блоков арифметических операций (сложение, вычитание, умножение, деление), вычисляется из введенных величин A, B и C по формуле T_V=(A+2C+AB)/5.

Тестирование в ручном режиме, по сравнению с автоматическим, дает больше возможностей и вариантов для более полной и всесторонней проверки работы блоков. Однако здесь потребуются некоторые знания и навыки оператора, понимание основ работы в среде, а также займет больше времени.

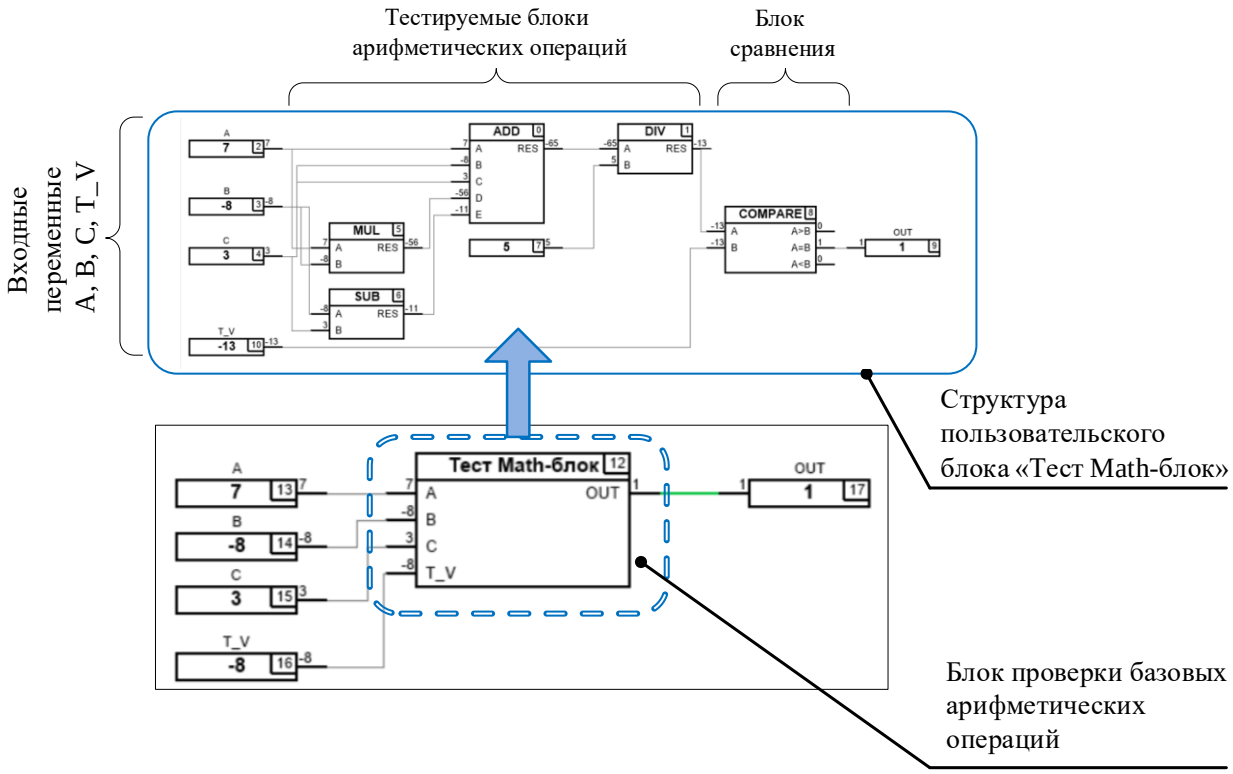



**Рисунок 3.9 – Пример ручного тестирования арифметических операций**

**Таблица 3.2 – Пример создания тест-кейсов для «Тест Math-блок» (фрагмент)**

| **Тест Math-блок** | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **A** | 7 | -30 | -100 | -498 | -326 | -561 | 0 | 513 | 78 | -54 | -62 |
| **B** | -8 | -18 | -154 | -261 | 198 | -323 | 0 | 769 | 26 | -39 | 33 |
| **C** | 3 | -65 | -657 | -1000 | 547 | 911 | 0 | -401 | 91 | -77 | 14 |
| **T_V** | -8 | 76 | 2797 | 25496 | -12756 | 36492 | 0 | 78841 | 457 | 379 | -416 |

## 3.3 Математические подходы обнаружения ошибок в программном обеспечении

Приступая к процедуре тестирования важно понимать, что данный процесс может занимать до 40-50% времени всего жизненного цикла проектируемой системы. Затраты (временные, финансовые, информационные), выделяемые на данный этап, могут быть достаточно значительными и не

признающими экономии, так как речь идет, во-первых, о качестве выпускаемого продукта, а, во-вторых, об имидже копании-производителя. Однако, не стоит впадать в крайность и соглашаться на совершенно любые меры и расходы в отношении этапа тестирования. Существуют математически точные методы, позволяющие сделать продолжительность и трудоемкость процесса тестирования целесообразной и достаточной для достижения требуемого уровня безошибочности. При этом стоит обратить внимание на то, что применение ниже приведенных критериев возможно лишь в случае тщательного документировании всех проводимых испытаний и их результатов. Выбор инструментов для ведения журналов ошибок в данном случае не имеет значения. Главное – чтобы не было утери информации, чтобы были сохранения все условия проведения тестов, обнаруженные ошибки, дефекты и нерегулярные ситуации при минимальных трудозатратах.

### 3.3.1 Критерий интенсивности обнаружения ошибок

Если считать, что во время одного эксперимента обнаруживается не более одной ошибки и каждая ошибка до начала следующего эксперимента устраняется, то можно предположить, что при благоприятном ходе отладки и испытания кривая зависимости выражается уравнением:

$$N(K) = 1 — п/K, \quad (3.1)$$

где $n$ — количество обнаруженных и устраненных ошибок;

$K$ — количество экспериментов.

Как видно из уравнения, данная кривая будет асимптотически стремиться к единице. Причем, чем меньше величина коэффициента n, тем быстрее функция N(K) будет стремиться к 1, а чем больше значение переменной К, тем значение

функции N(K) ближе к 1 (Рисунок 3.10, кривые 1, 2, 3). Кривая 4 свидетельствует о неблагополучном ходе процесса [53].

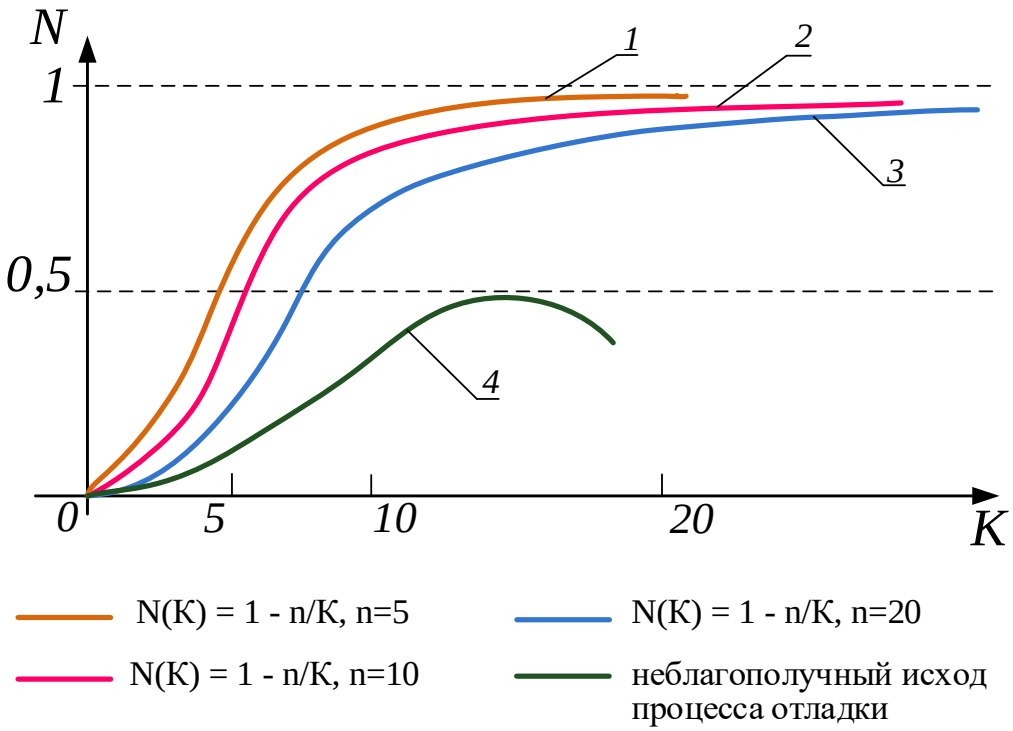



**Рисунок 3.10 – Кривая зависимости прекращения испытаний от количества обнаруженных и устраненных ошибок**

Тогда в качестве критерия прекращения испытаний можно принять следующее условие: $N > 0{,}95$ (рекомендуемые значения находятся в интервале 0,9-0,97) при обнаружении в последних двухстах экспериментах не более трех несущественных ошибок [53].

Критерий интенсивности обнаружения ошибка основан на статической модели надежности и не учитывает время появления ошибок. Поэтому его применение будет наиболее удобно при проведении тестирования в ручном режиме. Данные критерий позволяет определить порядковый номер тест-кейса, после которого будет достигнет заданный уровень проверенности программного продукта, и испытания можно прекратить.

### 3.3.2 Критерий заданного значения средней наработки на отказ

В рамках применения данного критерия, который также носит название *критерий Дж. Д. Муса*, сделано два предположения:

1. «Суммарное количество обнаруженных и устраненных дефектов в программе (под дефектом понимается любая причина неудовлетворенности свойствами программы) описывается показательной функцией времени функционирования τ» [53]:

$$n = N_0[1 - \exp(-\frac{C \cdot \tau}{M_0 T_0})], \quad (3.2)$$

где $N_0$ — исходное количество дефектов в программе;

$M_0$ — общее количество дефектов, которое может проявиться за время эксплуатации программного средства;

$T_0$ — средняя наработка на отказ в начале испытаний[1], час;

$C$ — коэффициент сжатия тестов.

Коэффициент $C \neq 1$ тогда, когда абсолютная реактивность программы при прогоне тестов или статистических испытаниях отличается от абсолютной реактивности при работе программы в реальных условиях. Если, например, за один час испытаний моделируется управляемый процесс, происходящий в реальных условиях в течение десяти часов, то коэффициент сжатия $C$ принимается равным 10 [53].

2. «Скорость обнаружения и устранения дефектов, измеряемая относительно времени функционирования программы, пропорциональна интенсивности отказов. Коэффициент пропорциональности $B=n/m$ называется *коэффициентом уменьшения дефектов*» [53]. Модель Муса относят к динамическим моделям непрерывного времени.

[1] Средняя продолжительность работы программного продукта между отказами. Для программных продуктов обычно подразумевается срок до полного перезапуска программы или полной перезагрузки операционной системы. Измеряется статистически, путём тестирования множества программ, или вычисляется методами теории надёжности.

Количество зарегистрированных отказов $m$ зависит от суммарного времени функционирования программы следующим образом:

$$m = M_0[1 - \exp(-\frac{C \cdot \tau}{M_0 T_0})], \quad (3.3)$$

(см. график зависимости на Рисунке 3.11)

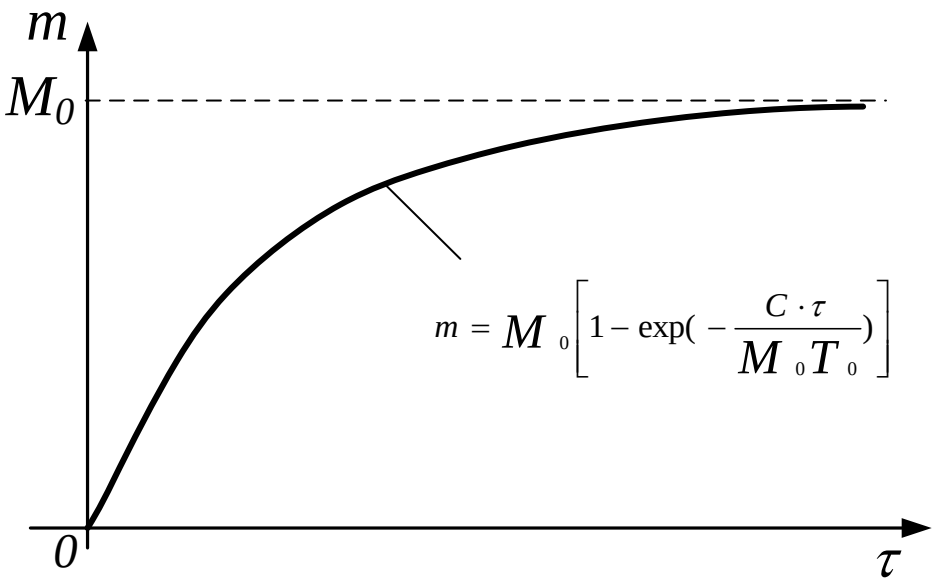



**Рисунок 3.11 – Зависимость количества отказов от времени функционирования системы логического управления**

«Модель Муса относят к динамическим моделям непрерывного времени. Это значит, что в процессе тестирования фиксируется время выполнения программы (тестового прогона) до очередного отказа» [5]. Его использование наиболее актуально при проведении тестирования в автоматическом режиме, когда необходимо вычислить оставшееся время для тестовых итераций, после достижения которого проверенный программный продукт будет иметь заданную величину наработки на отказ.

Значение средней наработки на отказ также зависит от суммарного времени функционирования:

$$T = T_0 \exp\left(\frac{C \cdot \tau}{M_0 T_0}\right), \quad (3.4)$$

(см. Рисунок 3.12)

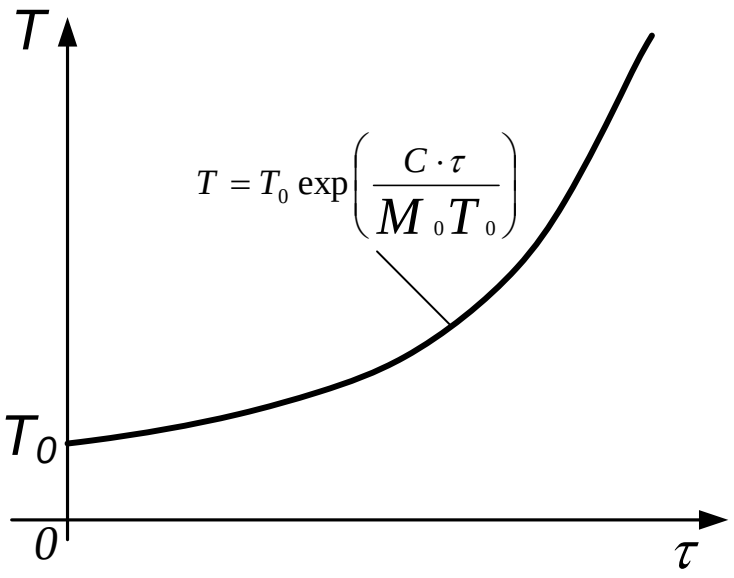



**Рисунок 3.12 – Зависимость текущей средней наработки на отказ от времени функционирования системы логического управления**

Если в ходе испытания обнаруженные ошибки устраняются, то текущее значение средней наработки на отказ будет увеличиваться. Таким образом, в качестве критерия завершенности испытания можно принять достижение требуемого (заданного) значения средней наработки на отказ $T_\tau$. Тогда, определяя периодически текущее значение средней наработки на отказ по этой формуле, можно при планировании дальнейшего хода испытания рассчитать требуемое время для дальнейшего прогона программы по формуле [53]:

$$\Delta\tau = \frac{M_0 T_0}{C} ln\left(\frac{T_\tau}{T}\right), \quad (3.5)$$

«При планировании отладки и испытания ПО следует учитывать влияние следующих факторов:

1) скорости выявления дефектов;

2) скорости устранения дефектов;

3) удовлетворенности машинным временем.

Первый фактор зависит от укомплектованности и квалификации испытателей, второй – от укомплектованности и квалификации группы программистов отладчиков, третий – от фондовооруженности (технической оснащенности) разрабатывающей (испытывающей) организации» [53].

### 3.3.3 Практические аспекты расчета математических критериев обнаружения ошибок в программном обеспечении

Рассмотрим описанные в пп. 3.3.1 - 3.3.2 теоретические положения для практических расчетов при проведении тестовых испытаний. Используем для этого процедуру проверки на наличие ошибок программы логического управления блока револьверной головки системы электроавтоматики станка СА535С10Ф4 – обрабатывающего центра наклонной компоновки с ЧПУ.

Данная программа написана на языке функциональных блоков и в качестве входных переменных имеет такие как: вращение барабана по часовой стрелке, вращение барабана против часовой стрелки, положение РГ в 0°, положение РГ в 90°, положение РГ в 180°, шпиндель работает, гнездо РГ свободно и пр. Выходные сигналы следующие: поместить инструмент в гнездо РГ, вращение барабана по часовой стрелке, вращение барабана против часовой стрелки и пр.

#### 3.3.3.1 Расчет и построение кривой критерия интенсивности обнаружения ошибок

Как было показано в п. 3.3.1 критерий интенсивности обнаружения ошибок выражается формулой (3.1) $N(K) = 1-n/K$, где n — суммарное количество обнаруженных и устраненных ошибок за число экспериментов К; К — количество экспериментов. Проведем несколько экспериментов и зафиксируем количество ошибок, обнаруженных в каждом из них. Результаты исследований приведены в Таблице. 3.3. Следует отметить, что рекомендуемы значения N находятся в

интервале от 0,9 до 1. В нашем случае в качестве критерия прекращения испытаний примем следующее условие: N > 0,9.

**Таблица 3.3 – Сводная таблица по количеству проведенных тестовых испытаний *К* и числу обнаруженных ошибок *n***

| **n=17** | | | | | | | **n=12** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **К** | 12 | 10 | 20 | 30 | 40 | 50 | 5 | 10 | 20 | 30 | 40 | 50 |
| **n/К** | 3,40 | 1,70 | 0,85 | 0,57 | 0,43 | 0,34 | 2,40 | 1,20 | 0,60 | 0,40 | 0,30 | 0,24 |
| **N(К)** | -2,40 | -0,70 | 0,15 | 0,43 | 0,57 | 0,66 | -1,40 | -0,20 | 0,40 | 0,60 | 0,70 | 0,76 |
| | | | | | | | | | | | | |
| **n=8** | | | | | | | **n=3** | | | | | |
| **К** | 5 | 10 | 20 | 30 | 40 | 50 | 5 | 10 | 20 | 30 | 40 | **50** |
| **n/К** | 1,60 | 0,80 | 0,40 | 0,27 | 0,20 | 0,16 | 0,600 | 0,300 | 0,150 | 0,100 | 0,075 | 0,060 |
| **N(К)** | -0,60 | 0,20 | 0,60 | 0,73 | 0,80 | 0,84 | 0,400 | 0,700 | 0,850 | 0,900 | 0,925 | **0,94** |

График зависимости полученных опытных данных представлен на рисунке 3.13.

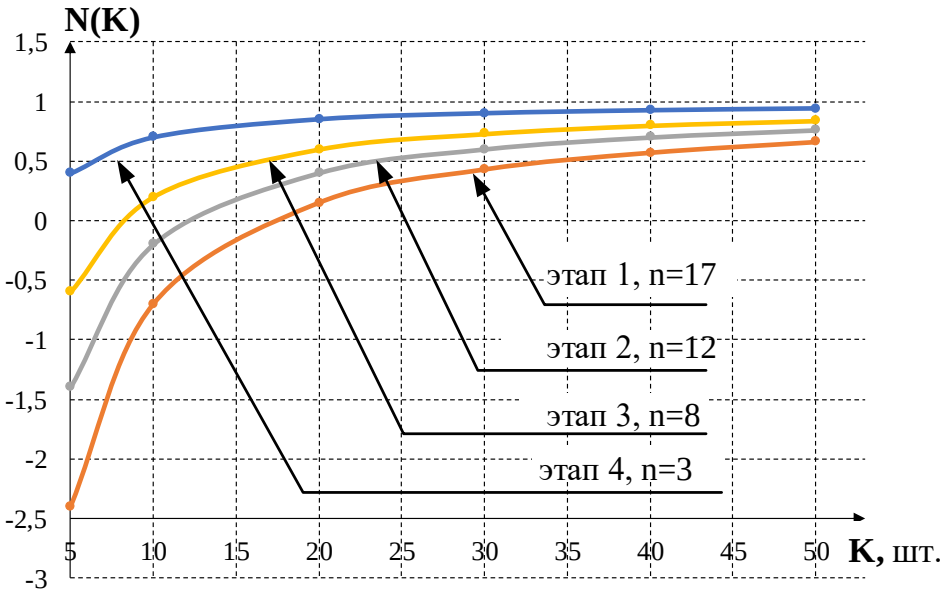



**Рисунок 3.13 – Экспериментальные кривые критерия интенсивности обнаружения ошибок при различных значениях количества обнаруженных ошибок *n***

Во время проведения тестирования последовательно на входные элементы подавались различные значения из тестового сценария. В процессе проверки количество обнаруживаемых ошибок *n* постепенно уменьшалось, т.е. интенсивность планомерно снижалась. После очередного тестового прогона были обнаружены и устранены 3 ошибки, после которых новых нерегулярных ситуаций зафиксировано не было. По формуле (3.1) можем рассчитать требуемое количество испытаний для получения заданного значения критерия, при условии, что за это время новых сбоев не появится.

Как видно на графиках рисунка 3.13, с уменьшением значения n кривая быстрее асимптотически стремится к 1, соответственно, чем больше n, тем достичь порогового значения критерия интенсивности обнаружения ошибок, принятого равным 0,9, сложнее и тем большее количество проверок К для этого потребуется. В нашем случае при n=3 на пятидесятом тестовом испытании значение критерия интенсивности, выраженного значением функции N(K), становится равным 0,94 (см. Таблицу 3.3). 0,94>09, следовательно, проверку работоспособности системы управления револьверной головкой при количестве испытаний, равном 50, по критерию интенсивности можно считать успешно завершенной.

#### 3.3.3.2 Расчет необходимого времени испытаний для достижения требуемого значения наработки на отказ

Значение средней наработки на отказ выражается формулой (3.4), а требуемое время для дальнейшего прогона программы по формуле (3.5).

Для расчета необходимого времени проверки программы для достижения заданной величины критерия завершенности $T_\tau$ зададим следующие значения аргументов формулы (3.5), выявленные экспериментально и на основе справочных данных [8, 11, 17, 18, 19, 20, 22]:

**Таблица 3.4 – Значения параметрических данных для расчета времени тестирования программы**

| | |
|---|---|
| $T_0 = 3$ | - средняя наработка на отказ в начале испытаний, час; |
| $C = 10$ | - коэффициент сжатия тестов; |
| $M_0 = 50$ | - общее количество дефектов, которое может проявиться за время эксплуатации программного средства; |
| $T_\tau = 1000$ | - требуемое (заданное) значение средней наработки на отказ, критерий завершенности испытаний, час; |
| $T = 8{,}15$ | - текущее значение средней наработки на отказ тестируемой программы, час; определяется по формуле (3.4) исходя из текущего момента времени испытаний τ=15 час (см. график на Рисунке 3.12.): $T=T_0 \exp\left(\frac{C\cdot\tau}{M_0 T_0}\right) = 3 * exp\left(\frac{10 * 3}{50 * 3}\right) = 8{,}15$ |

*Примечание*: Значение $T_0$ демонстрирует фактическое время безошибочного функционирования программы логического управления РГ в начале испытаний, т.е. когда она имеет существенное количество необработанных сбоев и нерегулярных ситуаций (неисправленных ошибок). Величина общего количества дефектов, которое может проявиться за время эксплуатации РГ $M_0$, была получена на основании анализа процедуры тестирования различных подобных РГ и количества обнаруженных в них ошибок группой тестировщиков и отладчиков [139, 141, 64, 24]. Величина требуемого значения средней наработки на отказ взята исходя из рекомендованных ГОСТом значений наработки на отказ для технических систем [19]. Расчетное значение текущей средней наработки на отказ получено исходя из текущего момента времени испытаний τ=15 час, т.е. на момент вычисления искомой $\Delta\tau$ тестовые прогоны управляющей программы продолжались 15 часов.

Тогда $\Delta\tau=\frac{M_0 T_0}{C} ln\left(\frac{T_\tau}{T}\right) = \frac{50*3}{10} ln\frac{1000}{8{,}15} =$ 72 часа.

Таким образом приведенные выше расчеты позволяют получить отлаженный программный продукт с показателем наработки на отказ в 1000 часов, при этом нет необходимости проверять на отказ программу все эти 1000 часов.

График функции T(τ), построенный по контрольным точкам (Таблица 3.5) будет выглядеть следующим образом (Рисунок 3.14):

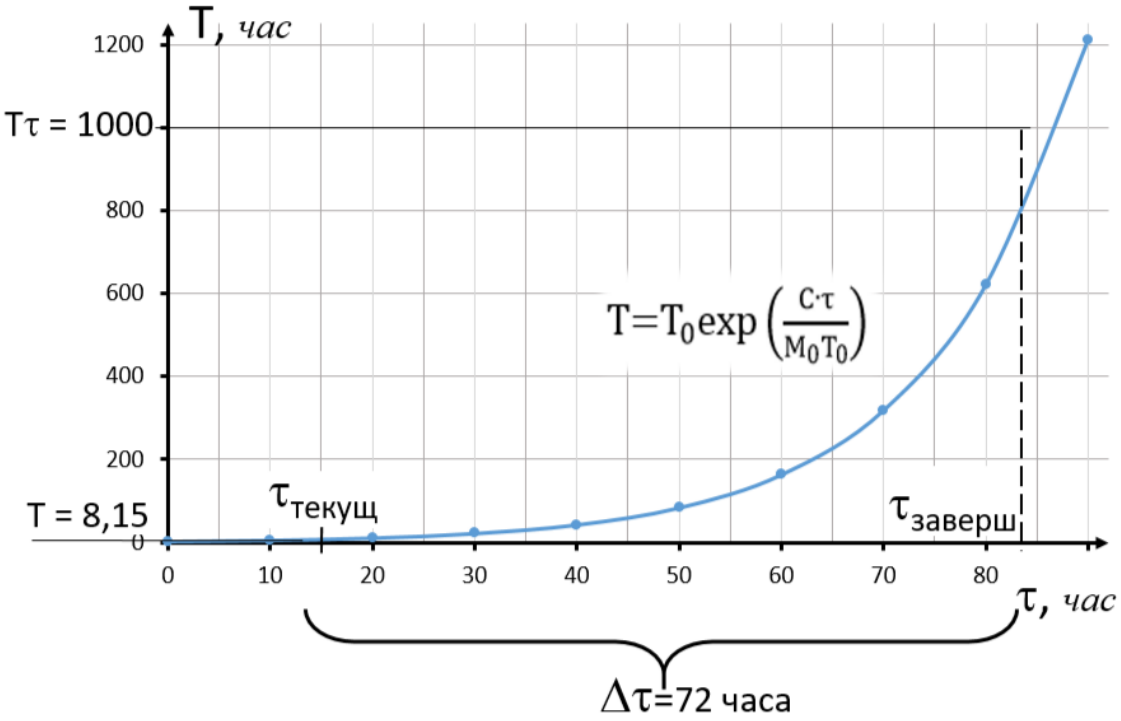



**Рисунок 3.14 – Экспериментальная кривая зависимости средней наработки на отказ от времени функционирования системы логического управления**

**Таблица 3.5 – Контрольные точки для построения графика зависимости средней наработки на отказ от времени функционирования**

| τ, час | 0 | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 |
|---|---|---|---|---|---|---|---|---|---|---|
| T, час | 3 | 6 | 11 | 22 | 43 | 84 | 164 | 319 | 621 | 1210 |

Следует отметить, что математический аппарат планирования испытаний, предлагаемый в данной работе, является вспомогательным инструментом. Использование описанных выше критериев не должно отвлекать тестировщиков непосредственно от процедуры тестирования, следования тестовому сценарию, фиксации результатов прохождения тест-кейсов, принятия решений о дальнейшем ходе проверки. Сам процесс проверки работоспособности программного продукта подразумевает обеспечение различных сред проведения экспериментов, различного программного окружения. Специалисты, привлекаемые к процессу

тестирования, должны иметь, главным образом, цель обнаружение ошибок, но не пытаться убедиться в их отсутствии. Тогда вспомогательный эффект от использования предлагаемых математических критериев будет наибольшим.

## 3.4 Выводы к главе 3

1. На основе сформированной в главе 2 структурной модели разработана методика тестирования систем логического управления, состоящая из шести последовательных шагов. Результат выполнения каждого шага является входными данными для последующего. Выполнение заключительного этапа методики позволяет получить протестированную систему управления и отчет об обнаруженных ошибках.

2. Описаны алгоритмы реализации функционального тестирования в ручном и автоматизированном режиме. Преимуществом автоматизированного тестирования является минимальное участие оператора в процессе тестирования, экономия времени. При этом нет необходимости в высокой квалификации оператора или владения какими-либо специальными знаниями. Тестирование в ручном режиме, по сравнению с автоматическим, дает больше возможностей и вариантов для более полной и всесторонней проверки работы блоков. Однако здесь потребуются некоторые знания и навыки оператора, понимание основ работы в среде, а также больше временных ресурсов.

3. В качестве примеров рассмотрено функциональное тестирование ядра программно реализованного логического контроллера в ручном и автоматизированном режимах.

4. Для планирования и организации тестовых испытаний, проводимых на основе предлагаемой методики, рассмотрены и предложены к применению математические критерии. Рассмотрены два математических подхода для обнаружения ошибок в программном обеспечении, их техническая база, а также приведены примеры их практического использования.

5. Результаты практических расчетов критериев обнаружения ошибок при проведении тестовых испытаний программы логического управления блока револьверной головки системы электроавтоматики станка СА535С10Ф4 показали, что:

- с уменьшением количества фиксируемых и отработанных ошибок отслеживаемый критерий быстрее стремится к 1; его заранее заданное пороговое значение, равное 0,9, достигается после 50го тестового прогона, когда значение критерия становится равным 0,94, что является сигналом к прекращению тестов; проверку работоспособности системы управления револьверной головкой по критерию интенсивности можно считать успешно завершенной;
- при работе с параметром наработки на отказ его заранее заданное значение достигается после времени испытаний, равного 72 часа; причем на начальном этапе в процессе проведения тестов ошибки обнаруживаются в бОльшем количестве, а по мере дальнейшего тестирования их число уменьшается, и чтобы обнаружить какие-либо сбои, группе тестировщиков приходится быть все изобретательнее в создании тест-кейсов; параллельно с этим группе специалистов, занимающихся исправлением обнаруженных сбоев и нерегулярных ситуаций, достается большой объем материала для работы в начале испытаний и значительно меньше – ближе к финалу проведения контрольных проверок.

# ГЛАВА 4. РЕАЛИЗАЦИЯ ПРИКЛАДНЫХ РЕШЕНИЙ ДЛЯ ТЕСТИРОВАНИЯ СИСТЕМ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ СТЕНДОВ

В рамках подготовки научно-квалификационной работы (диссертации) было принято решение о создании испытательных стендов для отладки работы системы логического управления станка СА535С10Ф4 – обрабатывающего центра наклонной компоновки с ЧПУ (Рисунок 4.1) и токарного станка с ЧПУ СА-700 (Рисунок 4.2).

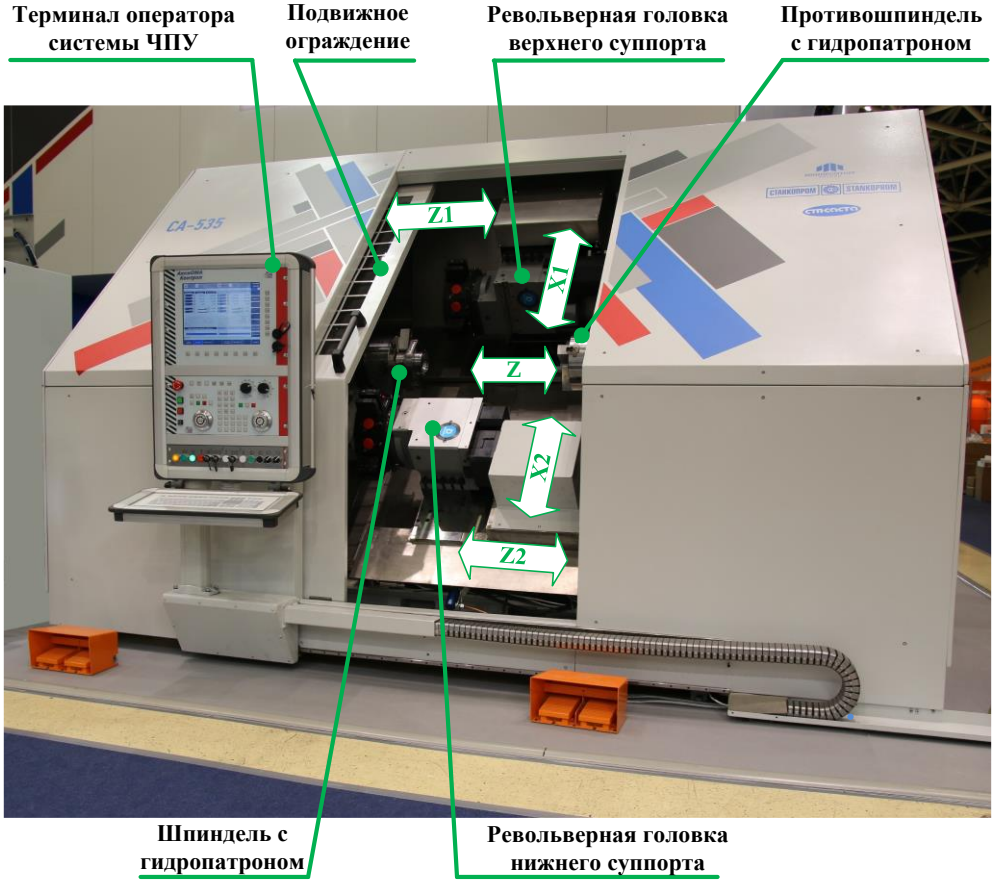



**Рисунок 4.1 – Обрабатывающий центр наклонной компоновки с ЧПУ СА535С10Ф4**

Станок СА535С10Ф4 предназначен для обработки деталей типа валов и фланцев сложной формы с высокой точностью. Кроме токарной обработки позволяет производить обработку гладких и резьбовых отверстий (торцевых несоосных и радиальных), фрезерование радиальных, торцевых, прямолинейных и

фасонных пазов и лысок. Допускается обработка деталей из чёрных и цветных металлов, из высоколегированных сталей, в том числе термообработанных. Наличие противошпинделя позволяет с перехватом произвести полную обработку детали с двух сторон без переустановки. Нижний суппорт даёт дополнительные возможности обработки и повышает производительность. Как видно из данной краткой характеристики, данный станок представляет собой конструктивно и кинематически сложное устройство, имеющее разветвленную структуру системы управления, обладающее широкой линейкой устройств электроавтоматики.

«Токарный станок СА-700 предназначен для токарной обработки в полуавтоматическом режиме наружных и внутренних поверхностей деталей типа тел вращения со ступенчатым и криволинейным профилем различной сложности в мелкосерийном и серийном производстве. Электроавтоматика управляет следующими узлами станка: шпиндель, система безопасности, освещение, револьверная головка, пиноль задней бабки, СОЖ. Имитационные модели данных элементов предполагается включить в разрабатываемый стенд» [87].


**Рисунок 4.2 – Токарный станок с ЧПУ СА-700**

Воспользуемся разработанными и описанными в предыдущих главах методикой и алгоритмами тестирования применительно к обрабатывающему центру СА535С10Ф4 и станку СА-700 и проверим на работоспособность программы логического управления, имеющиеся в них.

## 4.1 Анализ структуры системы логического управления

**Шаг 1 методики.** Согласно методике стендового тестирования систем логического управления, показанной в главе 3, на первом этапе проверки СЛУ осуществляется анализ ее структуры, изучение принципиальной электрической схемы, функциональных возможностей системы. Для реализации этого шага была тщательно изучена документация станков СА535С10Ф4 и СА-700, в особенности их принципиальные электрические схемы, дающие наибольше информации о параметрах систем электроавтоматики.

Подробное изучение необходимых для этих двух станков входных и выходных сигналов, используемых в работе устройств, позволило прийти к выводу о том, что один из станков – СА-700 – относится к группе микро СЛУ (15-128 точек ввода/вывода, Рисунок 3.2), имеющем в своем арсенале не более 100 входов/выходов, а СА535С10Ф4 – к группе микро СЛУ с числом входов/выходов больше 100. Отсюда следует, что для обрабатывающего центра разработка стенда тестирования, полностью повторяющего все входы и выходы, будет нецелесообразной. В соответствии с этим в рамках реализации следующего шага процедуры стендового тестирования применим разделение всех элементов системы электроавтоматики обрабатывающего центра на условно независимые блоки (модули). Чтобы это стало возможным, необходима процедура формирования этих самых модулей для системы автоматизации станка.

Что касается стенда для станка СА-700, то в этом случае он может полностью имитировать работу самой проверяемой системы. Однако для удобства также проведем в токарном станке с ЧПУ группировку всех имеющихся элементов на

блоки. Это не займет много времени в связи с их малочисленностью, но сделает пользование готовым стендом тестирования намного удобнее.

## 4.2 Выделение структуры модулей в программе логического управления электроавтоматикой обрабатывающего центра СА535С10Ф4 и токарного станка с ЧПУ СА-700

**Шаг 2 методики.** При формировании модулей учитывался принцип разделения по функциональному признаку, т.е. в один блок объединялись функционально близкие элементы. В результате такого разбиения для станка СА535С10Ф4 получились следующие некоторым образом обособленные группы элементов:

- Сигналы системы питания станка (блок «Питание») включают в себя:

группы элементов, отвечающих за систему питания станка, контроль фаз, срабатывание аварийного выключения, а также питание самого стенда, соответствующих световых индикаторов;

- Управляющие сигналы револьверной головки (РГ, блок «Револьверная головка») включают в себя:

группу элементов, отвечающих за управляющие сигналы на револьверную головку, т.е. имитация срабатывания предохранительной муфты, привода вращения инструмента, фиксации/расфиксации диска РГ, положения в нуле РГ, световые индикаторы, отображающие включение или выключение зажима диска, соответствующие световые индикаторы (станок СА535С10Ф4 имеет в своей комплектации револьверные головки верхнего и нижнего суппортов, поэтому тестирование каждой РГ будем производить поочередно);

- Сигналы шпинделя станка (блок «Шпиндель») включают в себя:

группу элементов, отвечающих за управляющие сигналы на шпиндель станка, такие как ограничение хода кулачков от центра и к центру, контроль давления в гидросистеме патрона, тип детали (вал/кольцо), педали зажима и разжима патрона, а также лампочки-индикаторы сведения/разведения кулачков и

контроль зажима патрона, соответствующие световые индикаторы (станок СА535С10Ф4 имеет в своей комплектации шпиндель и противошпиндель, поэтому тестирование каждого их этих узлов будем производить поочередно);

– Управление транспортером стружки (ТС, блок «Отвод стружки») включает в себя:

группу элементов, отвечающих за управление транспортером стружки, а именно: имитация движения ТС вперед и назад, проверка активности СОЖ, соответствующие световые индикаторы;

– Управление гидростанцией (блок «Гидростанция») включает в себя:

группу элементов, отвечающих за управление гидростанцией, такие как проверка уровня масля не ниже минимального, включение двигателя подачи масла, включение вентилятора теплообменника, контакт термостата теплообменника, соответствующие световые индикаторы;

– Система охлаждения инструмента (блок «СОЖ») включает в себя:

группу элементов, отвечающих за обеспечение рабочего уровня смазочно-охлаждающей жидкости (СОЖ): имитация минимального и максимального уровня СОЖ, засоренности фильтрующего полотна и включения перемотки фильтрующего полотна, имитация включения двигателя подачи СОЖ и барабанного магнитного сепаратора, соответствующие световые индикаторы;

– Система безопасности в рабочей зоне станка (блок «Безопасность») включает в себя:

группу элементов, отвечающих за систему безопасности в рабочей зоне станка: имитация открытия/закрытия дверей облицовки, а также левой и правой дверей заграждения, соответствующие световые индикаторы;

– Управление станцией охлаждения станка (блок «Станция охлаждения») включает в себя:

группу элементов, отвечающих за управление станцией охлаждения станка: имитация включения/выключения двигателя насоса, контроль уровня рабочей жидкости и засоренности фильтра рабочей жидкости, а также соответствующие световые индикаторы;

– Управляющие сигналы координатных осей станка (блок «Оси») включают в себя:

группу элементов, отвечающих за управляющие сигналы на координатные оси станка – имитация положения в нуле и аварийного позиционирования осей X1, X2, Y, Z, Z1, Z2;

– Опциональные входы/выходы (блок «Дополн. входы/выходы») включают в себя:

группу элементов, подключаемых опционально, т.е. в зависимости от изменяющихся требований объекта управления и условий функционирования.

В результате такой дифференциации всех возможных элементов системы электроавтоматики станка на подгруппы, объединенные по выполняемым функциям, были получены 10 условно независимых блоков. Преобразованная в такой вид СЛУ становится приемлемой для применения стендового тестирования с использованием модульного принципа, а также для дальнейшего применения методики тестирования, описанной в главе 2.

В дальнейшем по данному макету предполагается сконструировать специализированный испытательный стенд для отладки системы управления обрабатывающего центра наклонной компоновки с ЧПУ СА535С10Ф4 для реализации стендового тестирования.

Токарный станок с ЧПУ СА-700 имеет более простую компоновку, чем обрабатывающий центр СА535С10Ф4, меньшее число узлов, исполнительных органов, а значит, более простую систему электроавтоматики. При создании стенда для проверки его работоспособности потребуется гораздо меньше элементов и времени. Анализируя и структурируя техническую документацию станка СА-700 были сформированы следующие блоки (группы) элементов:

– Освещение (блок «Освещение») включает в себя:

световой индикатор, обозначающий состояние системы освещения – включена либо выключена;

– СОЖ (блок «СОЖ») включает в себя:

световой индикатор, обозначающий состояние системы, осуществляющей смазочно-охлаждающие функции, – включена либо выключена;

- Шпиндель (блок «Шпиндель») включает в себя:

группу элементов, отвечающих за управляющие сигналы на шпиндель станка, такие как зажим и разжим шпинделя, а также лампочки-индикаторы сведения кулачков (зажим шпинделя), разведения кулачков (разжим шпинделя), вращение по и против часовой стрелки;

- Система безопасности (блок «Защитное ограждение») включает в себя:

группу элементов, осуществляющих подачу двух вариантов управляющих сигналов на движение защитного ограждения влево либо вправо, а также два световых индикатора соответствующих движений;

- Револьверная головка (блок «Револьверная головка») включает в себя:

группу элементов, отвечающих за управляющие сигналы с датчиков положения РГ (а их четыре), а также а также лампочки-индикаторы, позволяющие отследить направление вращения РГ (против или по часовой стрелке), а также включение/выключение тормоза, блокировку РГ и размагничивание тормоза;

- Пиноль задней бабки (блок «Пиноль») включает в себя:

группу элементов, отвечающих за движение пиноли вправо либо влево, а также два световых индикатора соответствующих движений.

## 4.3 Проектирование и реализация стенда тестирования

**Шаг 3 методики.** На этапе технического проектирования стенда, после разделения имеющейся СЛУ на модули, была разработана схема расположения элементов будущего испытательного стенда как на Рисунке 4.3. Испытательные стенды могут объединять в себе несколько частей в одном корпусе. Это позволяет одновременно контролировать большое число параметров проверяемого объекта.


**Рисунок 4.3 – Макет испытательного стенда после выделения структуры модулей в СЛУ**

Говоря о СЛУ станка «имеем в виду систему электроавтоматики обрабатывающего центра. Электроавтоматика подобного типа станков на 70% состоит из датчиков и исполнительных механизмов, работающих с дискретными сигналами, поэтому для моделирования сигнала датчиков были использованы тумблеры ON-OFF MTS-201 с фиксацией и кнопки без фиксации нажатия OFF-(ON) PB-05-7R-G, а для визуального контроля сигналов, передаваемых на исполнительные устройства, была использована светодиодная индикация (с применением светодиодов с держателем L-608G)» [46]. В соответствии с вышеизложенным такой комплекс систем контроля и диагностики можно назвать логическим, т.е. работающим с сигналами типа 0 или 1. Именно его и подвергнем процедуре тестирования.

На этапе проектирования стенда тестирования были разработаны проекты испытательных стендов применительно станкам СА-700 и СА535С10Ф4 (Рисунок 4.4, Рисунок 4.5).

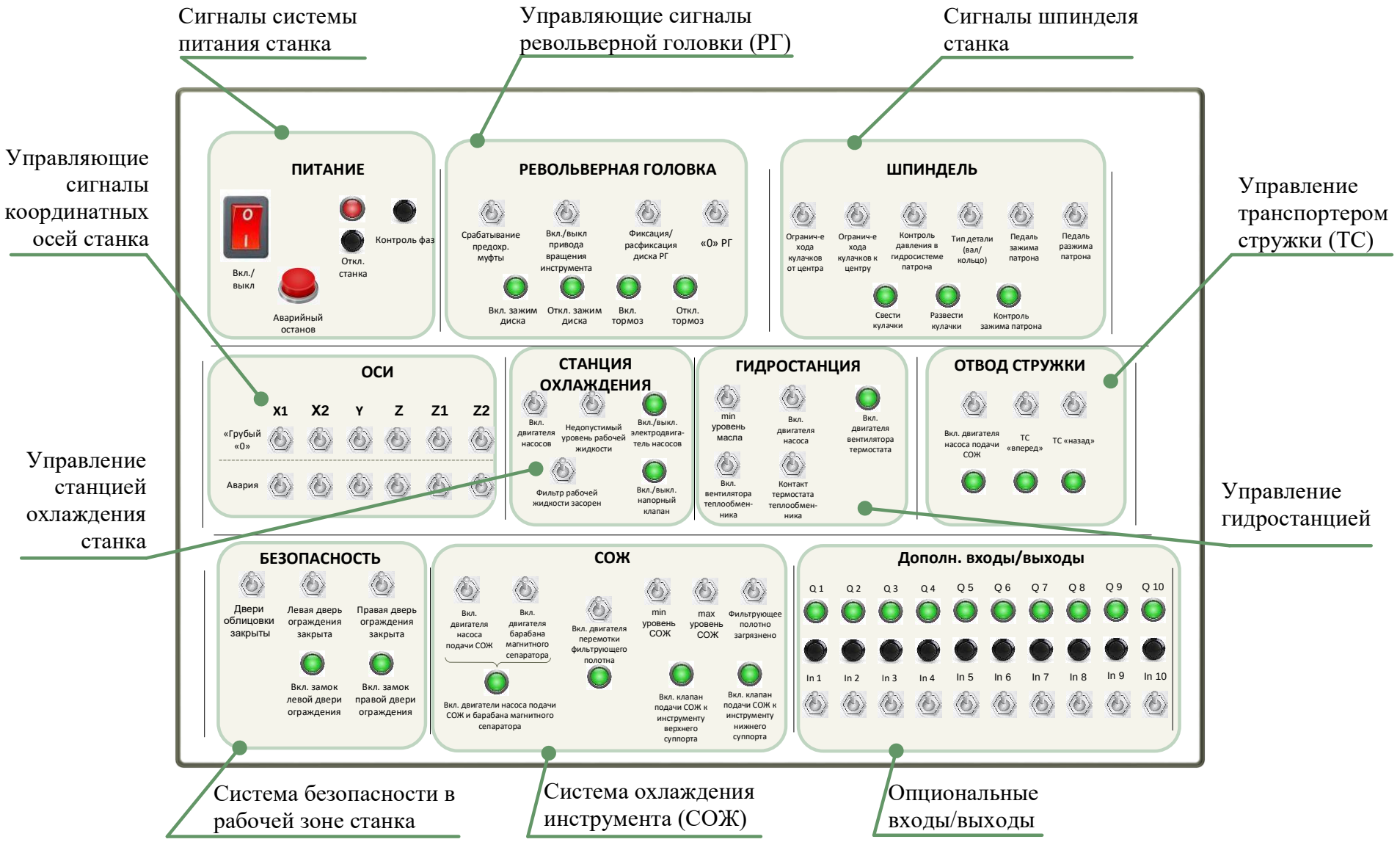



**Рисунок 4.4 – Внешний вид панели блока управления испытательного стенда для станка СА535С10Ф4**

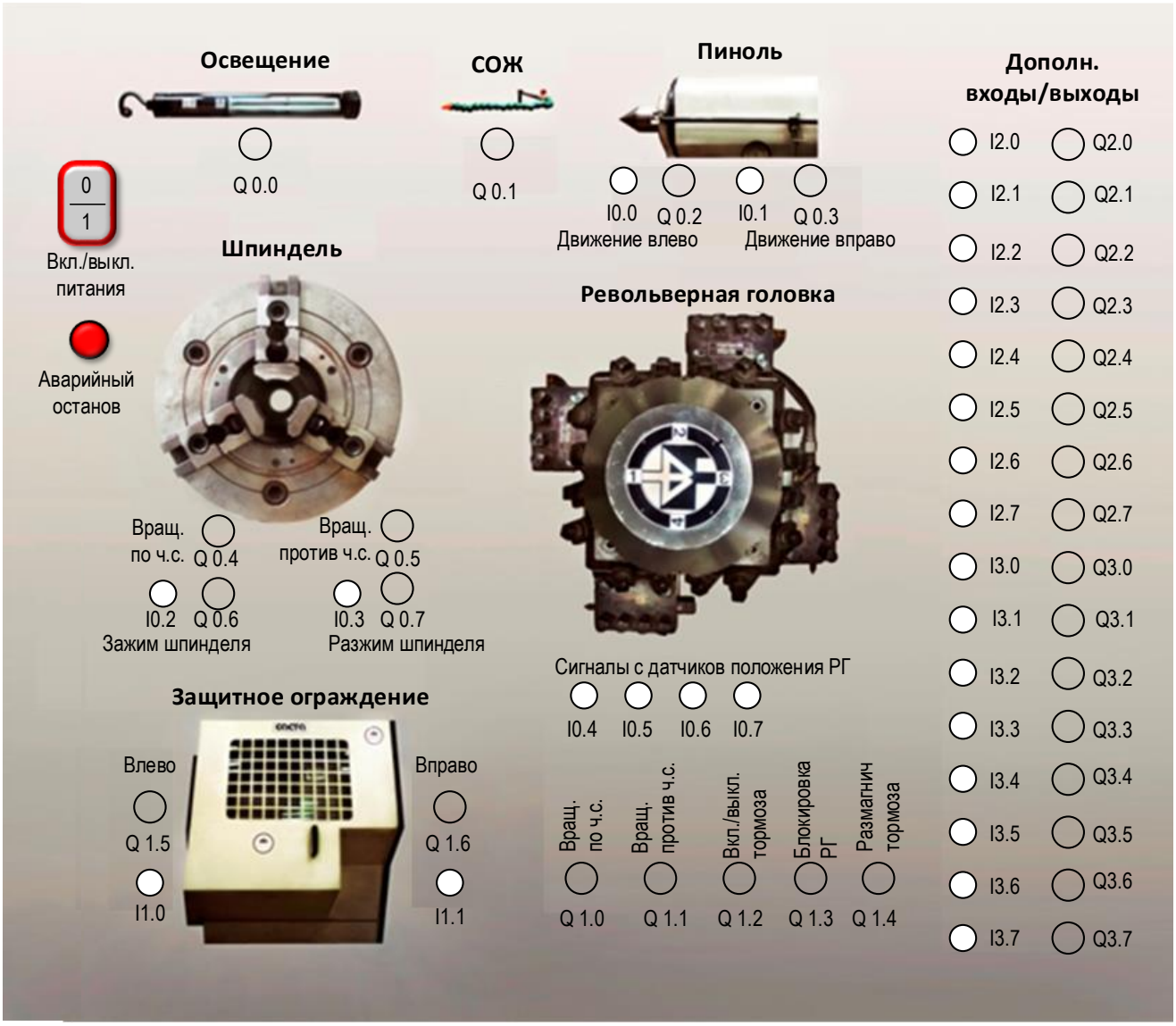



**Рисунок 4.5 – Внешний вид панели блока управления испытательного стенда для станка СА-700**

За основу в проектируемом стенде для обрабатывающего центра СА535С10Ф4 взято 43 постоянно подключенных цифровых входа (не считая элементов питания), 23 постоянно подключенных цифровых выходов, а также 10 подключаемых опционально цифровых входов и 10 выходов. В случае с токарным станком с ЧПУ СА-700 получили 10 постоянно подключенных цифровых входа (не считая элементов питания), 15 постоянно подключенных цифровых выходов, а также 16 подключаемых опционально цифровых входов и 16 выходов. Для лучшего понимания актуальности применения стендового тестирования были также оценены и проанализированы временные затраты на создание стенда (Таблица 4.1).

**Таблица 4.1 – Анализ времени, затрачиваемого на разработку и сборку стендов**

| Станок | Название модуля (блока) | Количество элементов, вынесенных на стенд, шт. | Трудоем-кость разработки, час |
|---|---|---|---|
| Обрабатывающий центр с ЧПУ СА535С10Ф4 | Питание | 5 (2 входа+3 выхода) | |
| | Револьверная головка | 6 (4 входа+2 выхода) | |
| | Шпиндель | 9 (6 входов+3 выхода) | |
| | Оси | 12 (12 входов) | |
| | Станция охлаждения | 5 (3 входа+2 выхода) | |
| | Гидростанция | 5 (4 входа+1 выход) | |
| | Отвод стружки | 6 (3 входа+3 выхода) | |
| | Безопасность | 5 (3 входа+2 выхода) | |
| | СОЖ | 10 (6 входов+4 выхода) | |
| | **Итого:** | **63 (43 входа+20 выхода)** | **~14,5** |
| | | | |
| Токарный станок с ЧПУ СА-700 | Шпиндель | 6 (2 входа + 4 выхода) | |
| | Система безопасности | 4 (2 входа + 2 выхода) | |
| | Освещение | 1 (1 выход) | |
| | Револьверная головка | 9 (4 входа + 5 выходов) | |
| | Пиноль задней бабки | 4 (2 входа + 2 выхода) | |
| | СОЖ | 1 (1 выход) | |
| | **Итого:** | 25 (10 входов + 15 выходов) | **~6,5** |

За основу для расчета приведенных в Таблице 4.1 величин времени были взяты данные, приведенные в типовых нормативах времени на работы по подготовке и проведению экспериментов на испытательных стендах [119], а также экспериментальные данные.

Разработанные стенды позволили смоделировать работу узлов при разработке и тестировании программ управления электроавтоматикой станка СА535С10Ф4 и СА-700. Для продолжения процесса далее необходимо спроектировать тестовые сценарии.

## 4.4 Создание тестовых сценариев

**Шаг 4 методики.** Для демонстрации практического применения 4 шага методики рассмотрим наиболее усложненный вариант системы электроавтоматики, имеющейся в станке СА535С10Ф4. Для стенда к станку СА-700 аналогичным образом составляются программные реализации станочных функциональностей.

Как видно на Рисунках 4.3, 4.4, разработанный испытательный стенд для проверки работоспособности системы электроавтоматики станка СА535С10Ф4 содержит 10 логических модулей, по количеству выделенных блоков в СЛУ обрабатывающего центра на этапе ее анализа. В рамках подготовки работы на стенде выполнялось помодульное функциональное и нагрузочное тестирования, в соответствии с решением, принятым в главе 1 (Таблица 1.5). Программа логического управления электроавтоматикой станка СА535С10Ф4 состоит из порядка 100 элементов (функциональных блоков среды программирования управляющих программ FBEditor и связей между ними) (Рисунок 4.6), из них 10 являются пользовательскими блоками, имеющими от одного до трех вложенных подпрограмм внутри себя.

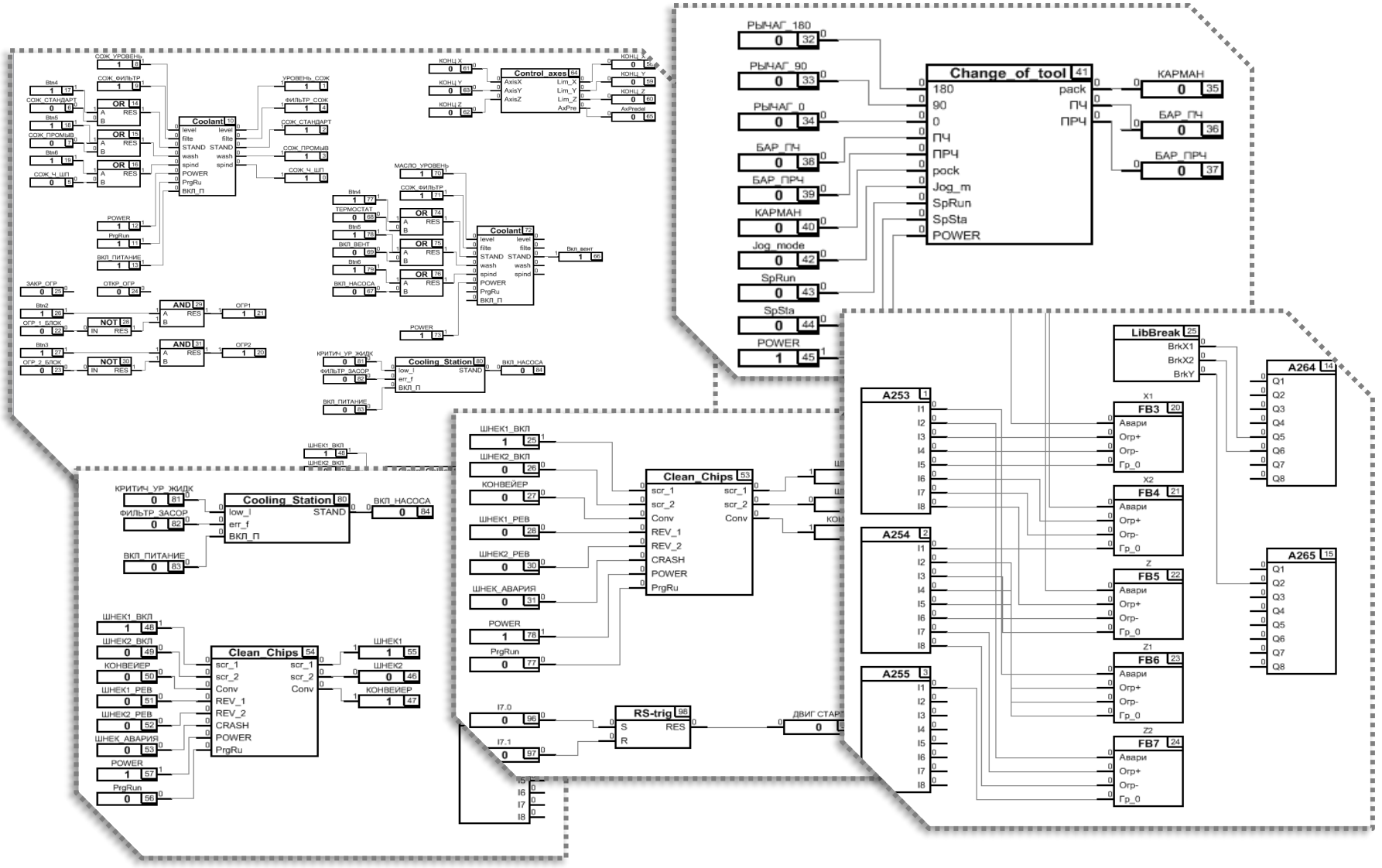



**Рисунок 4.6 – Фрагменты программы логического управления для тестирования на испытательном на стенде**

### 4.4.1 Подготовка тест-кейсов для функционального тестирования

В качестве примера, демонстрирующего процесс подготовки тестового сценария, рассмотрим тестирование модуля элементов, реализующих систему управления работы револьверной головки. На Рисунке 4.7 изображена схема функциональных элементов, отображающих ее работу в среде FBEditor. В нижней части Рисунка раскрыто содержание и логика работы программируемого пользовательского блока «Change_of_tool». Вся процедура проверки работы системы управления револьверной головкой сводится к подаче на входы «Change_of_tool» различных комбинаций нулей и единиц, имеющих физический смысл (положение/отсутствие рычага в позиции 0, 90, 180 градусов, вращение барабана по/против часовой стрелки, наличие/отсутствие свободного кармана для инструмента, включение ручного режима, вращение/останов шпинделя).

В таблице 4.1 представлен фрагмент тестового сценария для проверки работоспособности модуля револьверной головки.

**Таблица 4.2 – Фрагмент тестового сценария для проверки работоспособности системы логического управления револьверной головкой**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Входные сигналы** | Питание станка включено | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | **1** |
| | Вращение барабана по часовой стрелке | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | **0** |
| | Вращение барабана против часовой стрелки | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | **0** |
| | Положение РГ в 0° | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **0** |
| | Положение РГ в 90° | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | **1** |
| | Положение РГ в 180° | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | **0** |
| | Шпиндель работает | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | **0** |
| | Гнездо РГ свободно (Карман) | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | **1** |
| **Ожидаемые выходные** | Поместить инструмент в гнездо РГ (Карман) | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | **1** |
| | Вращение барабана по часовой стрелке | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | **0** |
| | Вращение барабана против часовой стрелки | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | **0** |

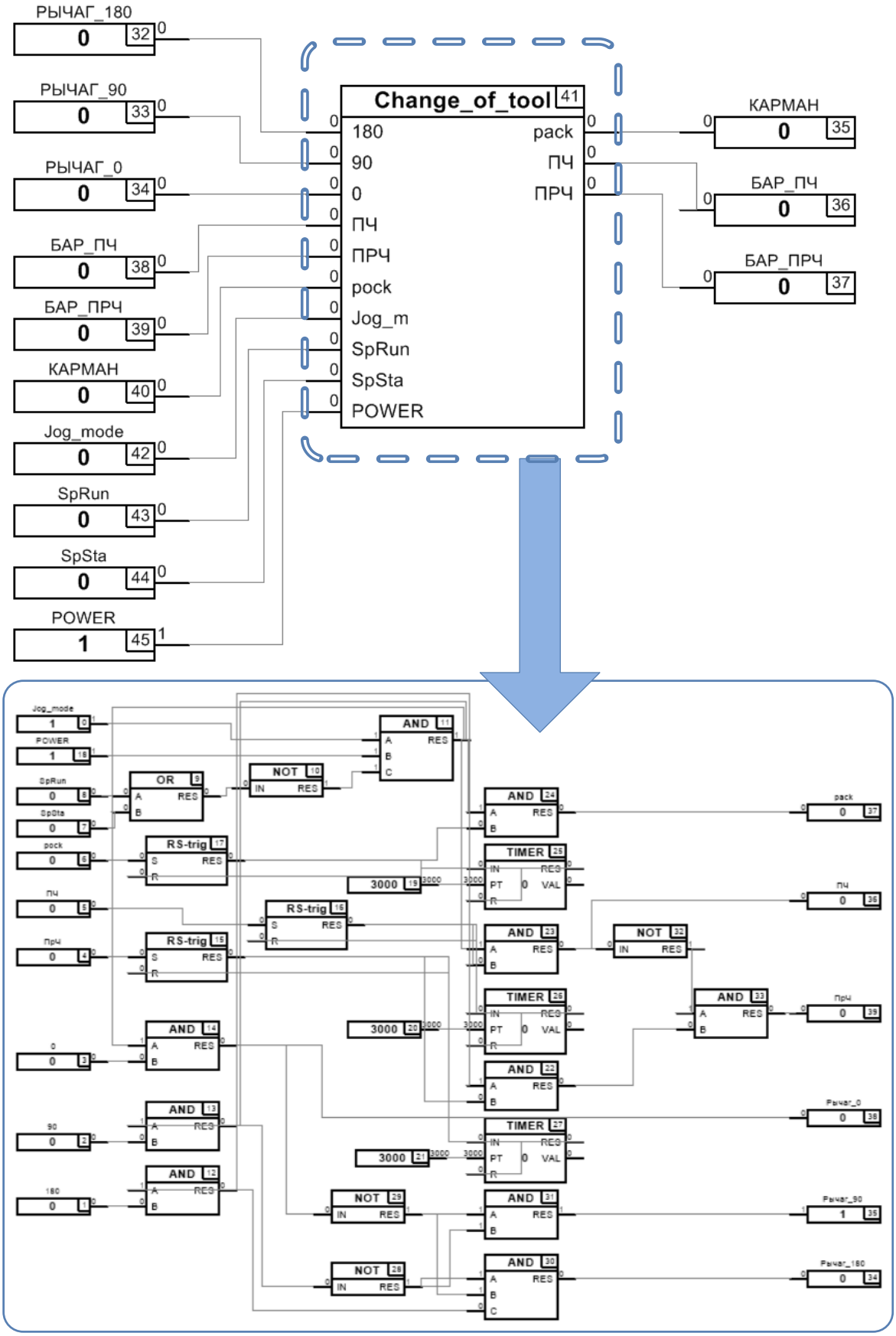



**Рисунок 4.7 – Программа логического управления блоком револьверной головки**

Как можно видеть, в данном наборе представлены различные комбинации подаваемых входных сигналов и соответствующие им требуемые (ожидаемые) выходные. Значение входного сигнала, равное нулю, означает разомкнутый контакт на испытательном стенде, т. е. выключенная соответствующая функция. Равенство входного сигнала единице означает замкнутый контакт на стенде, т.е. включенная соответствующая функция. Замкнутым и разомкнутым контактам соответствуют включенный либо выключенный тумблер. Срабатывание или несрабатывание каких-либо выходных сигналов фиксируется на панели управления стенда включением или выключением световых индикаторов (лампочек) соответственно. Таким образом осуществляется функциональное тестирование программы логического управления модуля револьверной головки.

Рассмотрим в качестве примера более подробно один тест-кейс из Таблицы 4.2 (в таблице выделен жирным). Приведенная комбинация нулей и единиц в качестве входных и ожидаемых выходных сигналов означает следующее. Питание на станок подано (1), и все его узлы находятся в рабочем состоянии. Вращение барабана револьверной головки ни по часовой (0), ни против часовой стрелки (0), не происходит, при этом револьверная головка находится в положении 90 град (1). Шпиндельный узел в неподвижном состоянии (0), гнездо револьверной головки свободно (1), ожидает принятия инструмента после его смены.

### 4.4.2 Выявление параметров для проведения нагрузочного тестирования

В главе 1 в Таблице 1.5. в качестве итогового решения, представляющего собой набор методов для тестирования СЛУ, кроме функционального тестирования, было выделено также и нагрузочное. Для уточнения напомним, что нагрузочное тестирование является одной из разновидностей нефункционального тестирования, относящегося к проверке на производительность. Данный метод проверяет способность системы сохранять работоспособность под воздействием большого количества обрабатываемых запросов, сохраняющихся достаточно длительное время.

Любое программное обеспечение должно работать под нагрузкой длительное время. Сбои и отказы системы могут привести к существенным убыткам, потере клиентов и другим неприятным последствиям. Именно поэтому, после тщательного изучения и анализа было выбрано нагрузочное тестирование. Оно является частью процесса тестирования производительности (способности системы работать под большими нагрузками). Нагрузочное тестирование дает возможность выявить максимальное количество однотипных задач, которые программа может выполнять параллельно. Данный вид тестирования позволяет определить, как и с какой скоростью работает программа под определенной нагрузкой.

«Обычно нагрузочное тестирование производится для многопользовательских систем с целью определения количества одновременно обслуживаемых клиентов без потери качества. В случае системы логического управления целесообразно оценивать зависимость используемых вычислительных ресурсов от сложности системы. Сложность системы может характеризовать несколько показателей, среди которых: количество аппаратных входов/выходов, количество используемых функциональных блоков в программе логического управления. Для расчета каждого блока в цикле управления расходуется часть вычислительных ресурсов, с ростом числа функциональных блоков увеличивается сложность программы и растет расход вычислительных ресурсов, поэтому для проведения нагрузочного тестирования системы логического управления целесообразно использовать указанный показатель в качестве базового» [88].

Нагрузочное тестирование выполнялось с использованием тестового набора управляющих программ (УП). Компоненты набора отличались количеством функциональных блоков в каждой УП. «Программы разрабатываются и загружаются в ядро контроллера автоматизации с использованием терминального клиента FBEditor, ранее разработанного на кафедре компьютерных систем управления ФГБОУ ВО «МГТУ «СТАНКИН»» [46]. В качестве системных показателей, которые будут отслеживаться во время нагрузочного тестирования, были выбраны следующие:

- ОЗУ, % – объем занимаемой суммарной ОЗУ (абсолютное значение расходуемой оперативной памяти);
- Терминал, % – объем используемой оперативной памяти, занимаемой терминалом;
- Ядро, % – объем используемой памяти, потребляемой ядром контроллера (приложение наиболее критичное к имеющимся вычислительным ресурсам);
- Время цикла, мс – время выполнения одного цикла программы логического управления; в первом приближении включает в себя: чтение входов, выполнение управляющей программы, установка выходов («является наиболее критичным показателем производительности системы логического управления, т.к. напрямую характеризует процессы реального времени» [88]).

В рамках подготовки научно-квалификационной работы (диссертации) было выполнено тестирование программ различного уровня сложности (с различным количеством компонентов) для определения зависимости параметров системы от количества функциональных блоков в обрабатываемой программе.

## 4.5 Проверка работоспособности пользовательских подпрограмм

**Шаг 5 методики.** На пятом этапе осуществляется проверка корректности работы пользовательских подпрограмм, библиотек и шаблонов в автоматизированном режиме. Этот шаг является важным и необходимым, так как любое программная среда для создания программ логического управления может претерпевать естественные фазы модернизации и обновления, что приводит к необходимости убедиться, что созданные ранее пользовательские шаблоны и библиотеки штатно работают в обновленной среде. Также необходимость проверки работоспособности пользовательских подпрограмм возникает при их использовании на различных аппаратных устройствах, при взаимодействии с различными операционными системами, нахождении в окружении других

различных блоков и связей. Все эти факторы могут существенно исказить работу пользовательских блоков, что неизбежно приведет к провалу всех испытаний технологического оборудования.

В рамках проверки работоспособности системы электроавтоматики обрабатывающего центра наклонной компоновки рассмотрим реализацию 5 шага методики на примере тестирования пользовательского блока булевой функции эквиваленции. Данная подпрограмма является элементом целой программы логического управления, используемым для проверки включения/выключения освещения, закрытия/открытия защитных ограждений и т.п. В соответствии с предлагаемой методикой тестирования необходимо предварительно проверить корректность работы данного пользовательского блока, как и других, ему подобных.

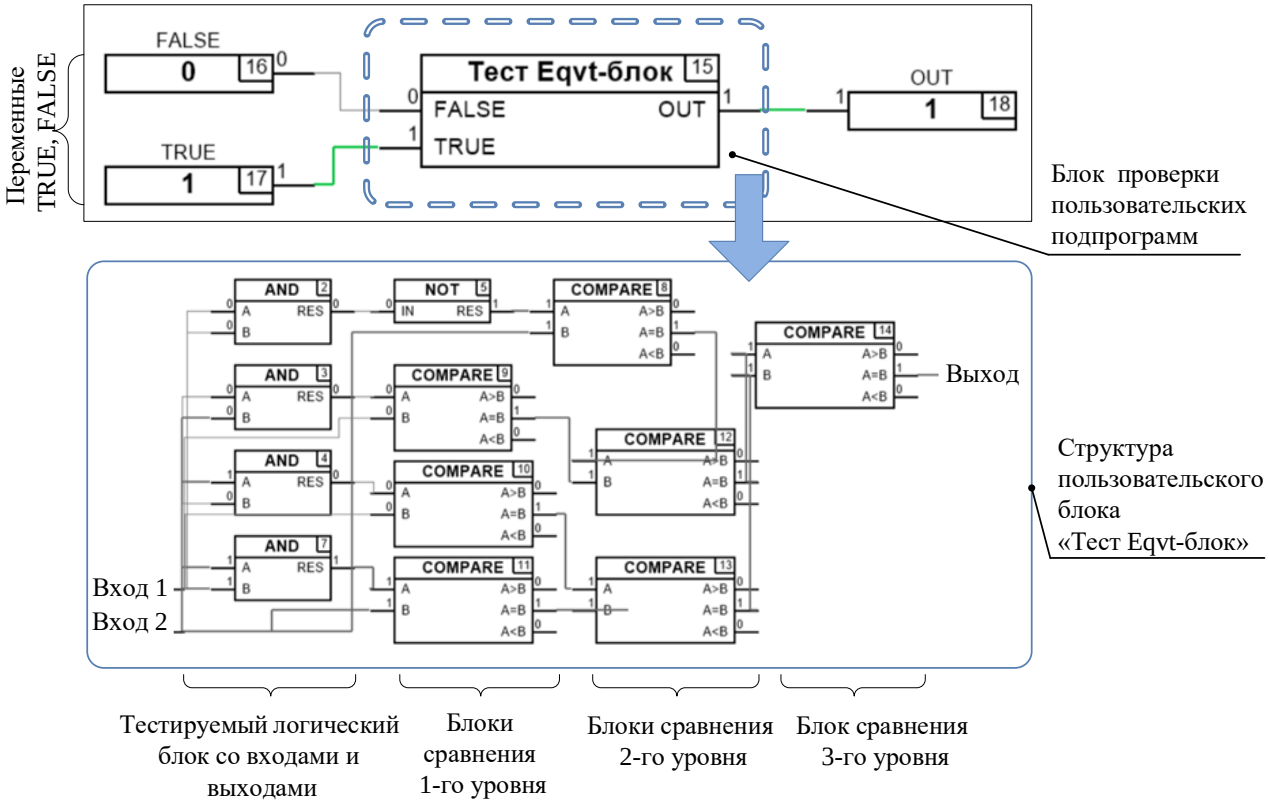



**Рисунок 4.8 – Пример тестирования пользовательских подпрограмм**

После успешного завершения всех проверок, касающихся пользовательских подпрограмм, можно приступать к осуществлению процедуры

тестирования всех блоков и модулей станка, вынесенных на специализированный стенд.

## 4.6 Проведение тестовых испытаний

Проведение тестовых испытаний – один из ключевых этапов методики тестирования систем логического управления технологическим оборудованием. Он подразумевает под собой проверку соответствия фактической работы проверяемой СЛУ ее ожидаемым результатам на конечном наборе тестов.

### 4.6.1 Функциональное тестирование

**Шаг 6 методики.** На основе сформированных тестовых сценариев, описанных выше, проводятся соответствующие испытания на проверку работоспособности различных модулей и систем станка.

Приведем фрагмент результатов функционального тестирования блока револьверной головки станка СА535С10Ф4.

В результате проведенного в соответствии с тестовым сценарием (Таблица 4.2) функционального тестирования был сформирован отчет об ошибках (Таблица 4.3). Из данного отчета можно видеть, что большая часть испытаний завершилась успешно (90%) и 10% тест-кейсов были не пройдены. Также в отчете обычно приводятся наиболее вероятные причины дефектов в логике работы управляющей программы, что является вспомогательной информацией для группы разработчиков, отвечающих за отладку и устранение обнаруженных ошибок.

Таблица является наглядным инструментом анализа результатов тестирования и предоставляет удобный формат данных для дальнейшей их проработки и исправления ошибок и уязвимостей.

**Таблица 4.3 – Отчет об ошибках, сформированный в результате тестирования системы управления револьверной головкой (фрагмент)**

| Тип сигнала | Физический смысл сигнала | Значение сигнала | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Входные сигналы | Питание станка включено | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | Вращение барабана по часовой стрелке | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Входные сигналы | Вращение барабана против часовой стрелки | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Положение РГ в 0° | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Положение РГ в 90° | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| | Положение РГ в 180° | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| | Шпиндель работает | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| | Гнездо РГ свободно (Карман) | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| Ожидаемые выходные сигналы | Поместить инструмент в гнездо РГ (Карман) | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| | Вращение барабана по часовой стрелке | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | Вращение барабана против часовой стрелки | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| Фактические выходные сигналы | Поместить инструмент в гнездо РГ (Карман) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 |
| | Вращение барабана по часовой стрелке | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| | Вращение барабана против часовой стрелки | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| | **Статус проверки:** | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| **Всего тестов – 20, Пройдено – 90%, Не пройдено – 10%** | | | | | | | | | | | | | | |

### 4.6.2 Нагрузочное тестирование

Выполнение нагрузочного тестирования является неотъемлемой частью тестирования промышленных систем автоматики. Предсказуемость

производительности системы и ее устойчивость при нагрузках, сопоставимых с реальными условиями использования, напрямую влияет на качество работы проектируемых технологических программных продуктов. Результаты такой проверки могут отличаться в зависимости от количества базовых компонент программы логического управления, числа входящих в нее циклов и связей между ними. Разница в результатах нагрузочного тестирования может быть также обусловлена типом выбранного контроллера и процессора. Все эти факторы имеют существенное влияние на способность проектируемой системы справляться с выполнением своих задач в условиях воздействия реальных нагрузок. Соответственно, изменение любого из перечисленных параметров приводит к необходимости выполнения нагрузочного тестирования СЛУ. В противном случае данный вид проверки можно не выполнять.

Результаты нагрузочного тестирования, полученные в данной работе с использованием некоторого вспомогательного ПО и проведенные для программ логического управления (далее – ПЛУ) с различным числом компонент, продемонстрированы в Таблице 4.4, а также в виде графиков на Рисунке 4.9.

**Таблица 4.4 – Результаты нагрузочного тестирования**

| **Результаты тестирования** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **N – количество элементов ПЛУ, шт.** | **100** | **1000** | **2000** | **5000** | **10 000** | **12 000** | **15 000** | **20 000** |
| **Время цикла, мс** | 0,8 | 1,4 | 2,6 | 7,2 | 18,5 | 25,1 | 30 | 35,3 |
| **Ядро, %** | 3,12 | 4,89 | 6,1 | 8,37 | 11,24 | 11,63 | 12,18 | 13,07 |
| **Терминал, %** | 4,18 | 8 | 11 | 31 | 34,5 | 35 | 36,3 | 37 |
| **ОЗУ, %** | 10,11 | 11,78 | 12,49 | 12,98 | 13,78 | 14,884 | 15,18 | 15,45 |

Нагрузочное тестирование проводилось для всех модулей, представленных на стенде (рис. 4.4). Различие заключалось в числе компонент среды программирования для составления программ логического управления.

Анализ результатов нагрузочного тестирования показывает, что с ростом сложности программы логического управления наблюдается рост значений всех

наблюдаемых показателей, но с разной скоростью. При количестве компонентов до 3000 линия графика выглядит достаточно пологой, но при этом фиксирует постепенное увеличение времени обработки сигналов программой. В случае если программа логического управления содержит более 3000 элементов, то кривая графика резко поднимается вверх, и время отработки цикла становится неприемлемо большим. Среднее количество элементов в реально работающих программах логического управления составляет 2-3 тыс., что дает возможность считать нагрузочное тестирование пройденным успешно.

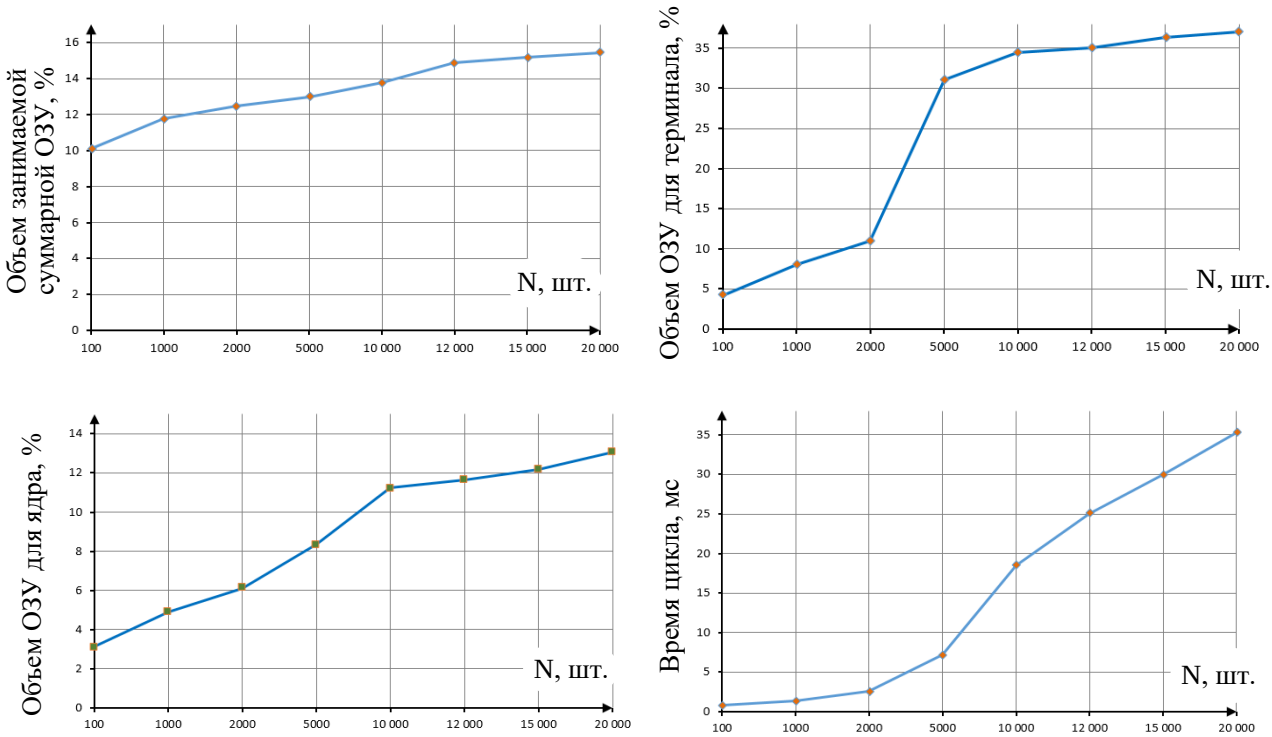



**Рисунок 4.9 – Зависимости используемой ОЗУ и времени цикла от количества элементов ПЛУ**

«Нагрузочное тестирование также показало, что тактовая частота процессора напрямую не влияет на размер потребляемых ресурсов процессора. Это говорит о том, что применяемые в системе аппаратные вычислительные устройства должны быть максимально сбалансированы с точки зрения всех характеристик производительности. Из этого можно сделать вывод, что наиболее существенным фактором, влияющим на потребляемые ресурсы процессора, является не только сложность алгоритмов программы логического управления, но и количество

ресурсов, привлекаемых самим приложением ядра для обслуживания внутренних потребностей» [88].

Следует также отметить, что загрузка центрального процессора (ЦП) может меняться в зависимости от занимаемых ресурсов операционной системы (ОС), установленных приложений окружений и т.д. В идеальном случае, для систем управления применяются ОС реального времени с ограниченной пользовательской функциональностью, позволяющие использовать всю мощь системы под нужды управления технологическим оборудованием. Еще одним фактором, влияющим на загрузку ЦП, является тип центрального процессора. В данном случае тестирование производилось на процессоре Intel® Core™ i3 4500U @ 1,8 GHz x 2 (PC) [46].

«Время цикла программы логического управления является, по сути, внутренним показателем системы управления и может варьироваться в зависимости от решаемой технологической задачи и применяемой платформы исполнения. В данной вариации также важна программная среда ОС. Для работы в режиме выполнения программы управления важно распределение потоков ОС, задание их приоритетов, обработки прерываний и т.д. На зависимость выполнения цикла программы управления влияет также показатель используемого процессора» [46].

## 4.7 Сравнительный анализ результатов применения предложенной методики

Для оценки времени, затрачиваемого на разработку СЛУ, был проведен анализ длительности разработки управляющих программ для ПЛК с использованием нормативных и справочных материалов [120, 123]. Итоговое время разработки включает в себя постановку задачи, выбор метода решения, создание алгоритма решения задачи, кодирование алгоритма одним из языков стандарта

МЭК 61131 [23], трансляцию и компиляцию программы, тестирование и отладка программы.

При определении времени, необходимого для разработки управляющей программы для ПЛК, учитываются множество различных факторов, вводимых в расчет с помощью поправочных коэффициентов. Перечислим основные критерии, учитываемые при расчете времени создания программного продукта:

- разрабатываемая программа является новой, то есть создается «с нуля», или разрабатывается на базе какой-либо существующей;
- стаж программиста;
- сложность используемых алгоритмов (алгоритмы стандартные, реализующие стандартные методики расчета, или алгоритмы оптимизации, моделирования систем и объектов);
- предполагается ли обработка больших объемов данных (например, статистические алгоритмы, алгоритмы поиска, задачи учета, отчетности);
- вид используемой входной информации (базы данных, обработка данных в режиме реального времени, переменная информация);
- объем входной информации;
- количество входов/выходов системы.

Применительно к управляющим программам в контроллерах, работающих в системах управления промышленным технологическим оборудованием, были получены [120, 123] величины времени разработки, представленные в Таблице 4.5.

Рассмотрим аналогичные параметры на примере анализа станка СА535С10Ф4 – обрабатывающего центра наклонной компоновки с ЧПУ и токарного станка с ЧПУ СА-700. Для сравнения проанализируем экспериментально полученные данные по продолжительности времени разработки управляющих программ данных станков при использовании стендов тестирования. Время, затрачиваемое на разработку самого стенда (5-20 часов в зависимости от количества элементов), входит во время, указанное в столбце результатов применения стенда на этапе тестирования (Таблица 4.6)

**Таблица 4.5 – Анализ времени, затрачиваемого на разработку различных СЛУ**

| Категория сложности УП | | Количество входов/выходов системы (шт.) | Время отладки разработки без стенда (час) |
|---|---|---|---|
| Количество элементов в УП (шт.) | Использование сложных алгоритмов (Да/Нет) | | |
| ~20-30 | Да | ~25 | 124 |
| | Нет | ~20 | 112 |
| ~50 | Да | ~40 | 485 |
| | Нет | ~30 | 410 |
| ~100 | Да | ~60 | 632 |
| | Нет | ~50 | 568 |
| ~200 | Да | ~110 | 986 |
| | Нет | ~100 | 873 |

В данной таблице предлагаются к рассмотрению количественные характеристики положительных эффектов, достигаемых за счет применения стендового тестирования, а значит и методики тестирования СЛУ в целом.

**Таблица 4.6 – Сравнительный анализ времени разработки программ логического управления**

| **Наименование станка** | **Название модуля (блока) на стенде** | **Количество элементов, вынесенных на стенд, шт.** | **Время разработки <u>БЕЗ</u> использования стенда, час** | ***Время разработки <u>С</u> использованием стенда, час** |
|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* |
| Обрабатывающий центр наклонной компоновки с ЧПУ СА535С10Ф4 | Питание | 5 (2 входа+3 выхода) | ~35 | ~30 |
| | Револьверная головка | 6 (4 входа+2 выхода) | ~50 | ~30 |
| | Шпиндель | 9 (6 входов+3 выхода) | ~100 | ~70 |
| | Оси | 12 (12 входов) | ~85 | ~65 |
| | Револьверная головка | 6 (4 входа+2 выхода) | ~50 | ~30 |
| | Шпиндель | 9 (6 входов+3 выхода) | ~100 | ~70 |

*Продолжение Таблицы 4.6*

| *1* | *2* | *3* | *4* | *5* |
|---|---|---|---|---|
| | Оси | 12 (12 входов) | ~85 | ~65 |
| Обрабатывающий центр наклонной компоновки с ЧПУ СА535С10Ф4 | Станция охлаждения | 5 (3 входа+2 выхода) | ~40 | ~30 |
| | Гидростанция | 5 (4 входа+1 выход) | ~50 | ~30 |
| | Отвод стружки | 6 (3 входа+6 выхода) | ~50 | ~33 |
| | Безопасность | 5 (3 входа+2 выхода) | ~40 | ~30 |
| | СОЖ | 10 (6 входов+4 выхода) | ~100 | ~75 |
| | | **Итого:** | **~550** | **~393** |
| | | | | |
| Токарный станок с ЧПУ СА-700 | Шпиндель | 6 (2 входа + 4 выхода) | ~20 | ~18 |
| | Система безопасности | 4 (2 входа + 2 выхода) | ~20 | ~18 |
| | Освещение | 1 (1 выход) | ~10 | ~8 |
| | Револьверная головка | 9 (4 входа + 5 выходов) | ~40 | ~37 |
| | Пиноль задней бабки | 4 (2 входа + 2 выхода) | ~20 | ~15 |
| | СОЖ | 1 (1 выход) | ~10 | ~8 |
| | | **Итого:** | **~120** | **~104** |

**Примечание*: включая время разработки и сборки стенда (5-20 часов в зависимости от количества элементов).

В Таблице 4.6 приведены временные затраты, необходимые для разработки управляющих программ, реализующих логику работы различных функциональных модулей станков СА535С10Ф4 и СА-700. Также указано итоговое суммарное время, необходимое для разработки программы логического управления для всех модулей (система электроавтоматики станков). Крайний правый столбец показывает соответствующие временные затраты, но уже при использовании стенда для проверки работоспособности управляющих программ систем электроавтоматики станков.

Величины времени, указанные в Таблице 4.6, были определены с учетом данных, полученных в ходе реализации программы логического управления станком СА535С10Ф4 и станком СА-700.

Анализируя представленные табличные данные можно видеть, что наиболее существенное сокращение времени происходит при использовании стенда для проверки работоспособности сравнительно сложных систем с числом входов/выходов и элементов управляющей программы больше 80. В таких случаях выигрыш во времени составляет до 30%. Применение специализированных испытательных стендов в более простых станках с более простой логикой работы выглядит чуть менее впечатляюще – порядка 10%, однако и это говорит о неоспоримом преимуществе стендового тестирования.

Как можно видеть из данных, представленных в Таблице 4.6, использование стенда при проведении процедуры тестирования дало выигрыш времени в 157 часов для обрабатывающее центра наклонной компоновки, что в процентном соотношении составляет 29%. Для токарного станка с ЧПУ сокращение времени при использовании стендового тестирования составило 16 часов – 13%. Такая разница в величинах временной выгоды объясняется разной степенью сложности двух рассматриваемых станков и их систем электроавтоматики соответственно. Для более сложного станка СА535С10Ф4 отказ от применения стенда на этапе проверки работоспособности системы управления электроавтоматикой существенно усложнит процесс контроля и приведет к значительным временным затратам. Станок СА-700 является более простым технологическим оборудованием, имеющем меньшее количество функциональных модулей в своем составе, значительно меньшее число контролируемых входов/выходов, более простую систему управления электроавтоматикой. Использование стенда для его отладки, бесспорно, облегчает процесс тестирования, но в меньшей степени, чем для сложных многокомпонентных станков.

Обобщая и систематизируя все вышесказанное, можно сделать следующие выводы. Наиболее существенное сокращение времени происходит при использовании стенда для проверки работоспособности сравнительно сложных

систем с числом входов/выходов и элементов управляющей программы больше 80. В таких случаях выигрыш во времени составляет до 30%. Применение специализированных испытательных стендов в более простых станках с более простой логикой работы выглядит чуть менее впечатляюще – порядка 10-13%, однако и это говорит о неоспоримом преимуществе стендового тестирования. Пользуясь полученными выводами можем дополнить Таблицу 4.5 еще одним столбцом, показывающим временные затраты на разработку управляющих программ при использовании стенда тестирования и наглядно демонстрирующим преимущество стендового тестирования с точки зрения временнОго фактора (Таблица 4.7).

**Таблица 4.7 – Анализ времени, затрачиваемого на разработку различных СЛУ, при использовании стенда и без него**

| Категория сложности УП | | Количество входов/выходов системы (шт.) | Время разработки без стенда (час) | Время отладки при использовании стенда (час)* |
|---|---|---|---|---|
| Количество элементов в УП (шт.) | Использование сложных алгоритмов (Да/Нет) | | | |
| ~20-30 | Да | ~25 | 124 | 110 |
| | Нет | ~20 | 112 | 100 |
| ~50 | Да | ~40 | 485 | 360 |
| | Нет | ~30 | 410 | 308 |
| ~100 | Да | ~60 | 632 | 456 |
| | Нет | ~50 | 568 | 410 |
| ~200 | Да | ~110 | 986 | 690 |
| | Нет | ~100 | 873 | 611 |

## 4.8 Выводы к главе 4

1. Разработанная в главе 3 методика апробирована на обрабатывающем центре наклонной компоновки с ЧПУ на примере тестирования системы электроавтоматики обрабатывающего центра, 70% сигналов которой являются дискретными (логическими), а также на токарном станке с ЧПУ СА-700 и его системы электроавтоматики.

2. Выполнено комплексное тестирование системы электроавтоматики станков СА535С10Ф4 и СА-700. Подробно рассмотрены и составлены варианты тест-кейсов для проверки работоспособности системы управления револьверной головкой обрабатывающего центра. Полученные результаты испытаний представлены в виде графиков и таблиц.

3. Спроектированы и успешно апробированы специализированные испытательные стенды тестирования, применение которых позволяет устранить наибольшее число ошибок в системах логического управления, возникающих на начальной фазе разработки программного продукта и важных для обеспечения безопасности и безаварийной работы оборудования.

4. В результате сравнительного анализа продолжительности времени разработки управляющих программ для программируемых логических контроллеров были получены результаты, демонстрирующие эффективность использования стендов на этапе тестирования программы логического управления. При этом время разработки сокращается в среднем на 8-15% для простых систем (количество входов/выходов не более 80) и на 25-30% для сложных систем (количество входов/выходов более 80).

# ЗАКЛЮЧЕНИЕ

1. Диссертация является законченной научно-квалификационной работой, в которой изложены новые теоретически обоснованные методические решения и разработки, связанные с сокращением времени тестирования систем логического управления технологическим оборудованием за счет разработки моделей и алгоритмов с применением специализированных испытательных стендов, имеющие существенное значение для развития станкостроительной отрасли.

2. Разработано графическое описание управляемых компонент станка, оснащенного системой логического управления, в виде ориентированного графа, позволившее выявить и сформулировать отличительные свойства систем логического управления станочным оборудованием.

3. Проведенное исследование существующих способов тестирования позволило установить взаимосвязи между видом тестирования и типом проверяемой системы логического управления, что позволило сформировать итоговое комплексное решение для проверки работоспособности систем логического управления технологическим оборудованием.

4. Разработано формальное описание процессов и объектов жизненного цикла технологического оборудования и входящих в него подсистем. Выявлены и формализованы взаимосвязи подсистем и процессов с учетом среды функционирования.

5. На основе установленных взаимосвязей разработана структурная модель комплекса тестирования систем логического управления, основанная на применении стендового тестирования и позволяющая производить комплексную проверку СЛУ технологического оборудования.

6. Предложены алгоритмы и сценарии тестирования систем логического управления технологическим оборудованием, ориентированные на применение языка функциональных блоков, позволяющие осуществлять тестирование программно-математического обеспечения ядра систем логического управления.

7. Сформулирована методика тестирования систем логического управления, основанная на принципе разделения систем управления по структуре, что позволяет производить помодульное тестирование программного обеспечения и дает возможность сокращения времени разработки до 30%.

8. Предложено использование двух математических критериев окончания тестовых испытаний, используемых в комплексе мер по обнаружению ошибок в программных продуктах. Приведены примеры расчетов, демонстрирующих их эффективность и результативность.

9. Сформулированная в работе методика, алгоритмы и сценарии успешно апробированы в ООО «Станко техника» при реализации научно-исследовательских, опытно-конструкторских работ, а также в учебном процессе при подготовке инженерных кадров по направлению 15.03.04 «Автоматизация технологических процессов и производств», что подтверждается справками об использовании.

10. Полученные результаты рекомендованы к применению на предприятиях станкостроительного профиля, разрабатывающих металлообрабатывающие станки и комплексы с ЧПУ, а также в учебном процессе при подготовке инженерных кадров по направлению 15.03.04 «Автоматизация технологических процессов и производств».

# СПИСОК СОКРАЩЕНИЙ

| Аббревиатура | Расшифровка |
|---|---|
| bug report | отчет с результатами проведения процедуры тестирования |
| bus coupler | шинный расширитель, используется в комплексе с блоком пассивных входов/выходов |
| CNC | Computer Numeric Control – числовое программное управление |
| EtherCAT | один из промышленных протоколов связи, применяемый для соединения ядра системы управления с внешними устройствами (исполнительными механизмами, датчиками и пр.) |
| FBD | Function Block Diagram – язык функциональных блоков, один из пяти стандартных языков программирования ПЛК |
| FBEditor | программная среда для создания и редактирования управляющих программ для ПЛК и SoftPLC |
| RTM | Requirement Traceability Matrix – матрица отслеживания требований |
| SERCOS | один из промышленных протоколов связи, применяемый для соединения ядра системы управления с внешними устройствами (исполнительными механизмами, датчиками и пр.) |
| SERCOS III | один из промышленных протоколов связи, применяемый для соединения ядра системы управления с внешними устройствами (исполнительными механизмами, датчиками и пр.) |
| SoftPLC | программно реализованный логический контроллер |
| АСУ | автоматизированные системы управления |
| АСУП | автоматизированные системы управления предприятием |
| АСУТП | автоматизированная система управления технологическим процессом |

| Аббревиатура | Расшифровка |
|---|---|
| БД | база данных |
| ЖЦ | жизненный цикл |
| ИТ-компания | компания, основным направлением деятельности которой являются информационные технологии |
| ОС | операционная система |
| ПК | персональный компьютер |
| ПЛК | программируемый логический контроллер |
| ПЛУ | программа логического управления |
| ПО | программное обеспечение |
| РГ | револьверная головка |
| СЛУ | система логического управления |
| СОТС | смазочно-охлаждающие технологические средства |
| СОЖ | смазочно-охлаждающая жидкость |
| СУ | система управления |
| ТЗ | техническое задание |
| ТО | технологическое оборудование |
| ТС | транспортер стружки |

| **Аббревиатура** | **Расшифровка** |
|---|---|
| УП | управляющая программа |
| ЦП | центральный процессор |
| ЧПУ | числовое программное управление |
| ЭВМ | электронно-вычислительная машина |

# СПИСОК ЛИТЕРАТУРЫ

1. Абдуллаев, Р.А. Практические аспекты реализации управления разнородным технологическим оборудованием электроавтоматикой в системах ЧПУ / Р.А. Абдуллаев // Вестник МГТУ «Станкин», № 1 (24), 2013, с. 52-55.

2. Амблер, С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки / С. Амблер – СПб. : Питер, 2005

3. Баурина, С.Б. Современные технологические тренды развития станкостроения в России / С.Б. Баурина, Е.О. Савченко // Вестник Российского экономического университета имени Г.В. Плеханова. – 2019. – № 2 (104). – С. 81-92.

4. Бейзер, Б. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер – СПб.: Питер, 2004. – 318 с. с ил. // ISBN 5-94723-698-2

5. Библиотека онлайн Студбукс. Модель Муса [Электронный ресурс]. – Режим доступа: https://studbooks.net/2016378/informatika/model_musa (дата обращения: 12.03.2020).

6. Библия QA – Часть1 – URL: https://easyqa.github.io/part_1/index.html (дата обращения: 18.03.2018)

7. Библия QA v. 2.0 – URL: https://vladislaveremeev.github.io/ (дата обращения: 18.03.2018)

8. Благодатских, В. А. Стандартизация разработки программных средств / В.А. Благодатских, В.А. Волнин, К.Ф. Поскакалов. – М. : Финансы и статистика, 2005. – 288 с.

9. Блэк, Р. Ключевые процессы тестирования. Планирование, подготовка, проведение, совершенствование. / Р. Блэк. – Москва: Лори, 2006. – 544 с.

10. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джакобсон. – М. : ДМК Пресс, 2001.

11. Василенко, Н.В. Модели оценки надежности программного

обеспечения / Н.В. Василенко, В.А. Макаров // Вестник Новгородского гос. ун-та. – 2004. – № 28. – С. 126–132.

12. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем : учеб. / А.М. Вендров. – 2-е изд., перераб. и доп. – М. : Финансы и статистика, 2006.

13. Волкова, Г.Д. Концептуальное моделирование проектных задач: учеб. пособие / Г.Д. Волкова – М.: ФГБОУ ВО «МГТУ «СТАНКИН», 2015.- 117 с.:ил.

14. Гимадеев, М.Р. Подготовка управляющих программ для станков с ЧПУ на платформе Heidenhain / М.Р. Гимадеев, В.М. Давыдов, А.В. Никитенко, В.А. Стельмаков // Хабаровск.: ТОГУ, 2015. – 140 с.

15. ГОСТ 19.70190 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения / База нормативной документации rostest.info – [Электронный ресурс]. – Режим доступа: https://rostest.info/gost/001.001.080.050/gost-19.701-90/ (дата обращения: 31.01.2021).

16. ГОСТ 2.103-2013 Единая система конструкторской документации. Стадии разработки. / Под ред. Е.И. Мосура – М.: ИД «Юриспруденция», 2019 – с. 8.: ил., табл.

17. ГОСТ 24.701-86. Единая система стандартов автоматизированных систем управления. Надежность автоматизированных систем управления. Основные положения.

18. ГОСТ 27.002-89. Надежность в технике. Основные понятия. Термины и определения.

19. ГОСТ 27.402-95 Надежность в технике / Межгосударственный совет по стандартизации, метрологии и сертификации. – М.: ИПК Издательство стандартов, 1997. – 59 с.

20. ГОСТ 34.003-90. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. Термины и определения.

21. ГОСТ Р 15.301-2016 Система разработки и постановки продукции на

производство rosgosts.ru – [Электронный ресурс]. – Режим доступа: https://rosgosts.ru/file/gost/01/040/gost_r_15.301-2016.pdf (дата обращения: 31.01.2021).

22. ГОСТ Р ИСО/МЭК 9126-93. Информационная технология. Оценка программной продукции. Характеристика качества и руководство по их применению.

23. ГОСТ Р МЭК 61131-3-2016 Контроллеры программируемые. Часть 3. Языки программирования / Федеральное агентство по техническому регулированию и метрологии. – М.: Стандартинформ, 2016. – 279 с.

24. Григорьев, С.Н. Диагностика автоматизированного производства / С.Н. Григорьев, В.Д. Гурин, М.П. Козочкин, В.А. Кузовкин, Г.М. Мартинов, Ф.С. Сабиров, В.А. Синопальников, В.В. Филатов - М.: Машиностроение, 2011. - 600с.

25. Дастин, Эл. Автоматизированное тестирование программного обеспечения / Эл. Дастин, Дж. Рэшка, Дж. Пол // Изд. Лори, 2014. – 579 с. // ISBN 978-5-85582-318-9

26. Денисенко, В.В. Компьютерное управление технологическим процессом, экспериментом, оборудованием: учебное пособие / В.В. Денисенко – М.: Горячая Линия – Телеком, 2009. – 608 с.

27. Деркач, Е.В. Проверка работоспособности системы логического управления на примере функционального тестирования исполнительного ядра программно реализованного контроллера. // Материалы международной научно-практической конференции «Наука сегодня. Вызовы, перспективы и возможности», часть 1. Научный центр «Диспут». Вологда, 2019 – С. 37-43 // eLIBRARY ID: 41669348 // ISBN 978-5-907083-76-9 (Часть 1)

28. Деркач, Е.В. Функциональное тестирование исполнительного ядра программно реализованного контроллера на этапе разработки программной составляющей. // Труды Седьмой Всероссийской научной конференции с международным участием «Информационные технологии и системы». Под ред. Ю.С.Попкова, А.В.Мельникова, Научное электронное издание. Ханты-Мансийск, 2019, С. 88-93 // ISBN 978-5-6042173-7-5

29. Деркач Е.В., Нежметдинов Р.А. Методика стендового тестирования систем логического управления // Сборник научных статей по итогам IX международной научной конференции «Приоритетные направления инновационной деятельности в промышленности». – Казань, 29-30 сентября 2021 г. – НПП Медпромдеталь, ООО Газпром Трансгаз – С. 161-163 // ISBN 978-5-6046923-9-4

30. Деркач Е.В., Нежметдинов Р.А. Подход к проведению тестирования программ логического управления электроавтоматикой станков // Научный рецензируемый журнал. – М.: Вестник МГТУ «СТАНКИН», 2020. – №4(55) – С. 8-13.

31. Деркач Е.В., Нежметдинов Р.А. Практические аспекты проведения тестовых испытаний исполнительного ядра систем логического управления // Наука сегодня: вызовы и перспективы [Текст]: материалы международной научно-практической конференции, г. Вологда, 23 сентября 2020 г. – Вологда: ООО «Маркер», 2020, С. 38-41 // ISBN 978-5-907341-13-5 , УДК 001.1 ББК 60 Н34

32. Деркач Е.В., Нежметдинов Р.А. Практические аспекты проведения тестирования систем логического управления с использованием стендов // Материалы XIV всероссийской конференции с международным участием «Машиностроение: традиции и инновации (МТИ – 2021)». Сборник докладов. – М.: ФГБОУ ВО «МГТУ «СТАНКИН», 2021. – С. 135-143 // eLIBRARY ID: 47379716 // УДК: 002:621

33. Деркач Е.В., Нежметдинов Р.А. Практические аспекты тестирования программно реализованных контроллеров, применяемых при автоматизации экологических и энергетических систем // Производство. Технология. Экология – ПРОТЭК’20: сборник трудов Всероссийской молодѐжной научно-технической конференции с международным участием (г. Москва, (29 сентября – 1 октября 2020 г.) / под ред. проф. В. А. Аксѐнова, доц. Е. В. Бутримовой, проф. Л. Э. Шварцбурга. – Москва : ФГБОУ ВО «МГТУ «СТАНКИН», 2020, С. 186-191 // ISBN 978-5-7028-0643-3

34. Деркач Е.В., Нежметдинов Р.А. Разработка методики и алгоритмов

стендового тестирования систем логического управления // Научный рецензируемый журнал. – М.: Вестник МГТУ «СТАНКИН», 2021. – №4(59) – С. 13-18.

35. Жерелева, А.О. Тестирование на основе моделей / А.О. Жерелева // Хабр – сайт русскоязычных блогов на тему информационных технологий, бизнеса и интернета. - [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/vk/articles/506048/ (дата обращения: 31.01.2021).

36. Иванова, Г.С. Объектно-ориентированное программирование / Г.С. Иванова, Т.Н. Ничушкина, Е.К. Пугачев. – М. : Изд-во МГТУ им. Баумана, 2001.

37. Иванова, Г.С. Технология программирования / Г.С. Иванова. – М. : Изд-во МГТУ им. Баумана, 2002.

38. Ильюк, Д. Тестирование безопасности — выбираем нужное / Д. Ильюк // PC Week/RE №14 (869) 26 августа 2014 // Интернет ресурс – URL: https://www.itweek.ru/security/article/detail.php?ID=165460 (дата обращения: 10.11.2021)

39. Кабак, И.С. Математическая модель для прогнозирования и оценки надежности программного обеспечения / И.С. Кабак // Вестник МГТУ “Станкин”. -2014.-№ 1

40. Калбертсон, Р. Быстрое тестирование / Р. Калбертсон, Кр. Браун, Г. Кобб. : [Пер. с англ.] М. [и др.] : Серия института качества программного обеспечения - Вильямс, 2002., 383 с. с ил., табл. - ISBN 5-8459-0336-х

41. Канер, С. Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: Пер. с англ. / С. Канер, Дж. Фолк, Енг Кек Нгуен. – Москва: ДиаСофт, 2001. – 544 с.

42. Каштальян, И.А. Программирование и наладка станка с числовым программным управлением / И.А. Каштальян // Минск.: БНТУ, 2015. – 135 с.

43. Кватрани, Т. Rational Rose и UML. Визуальное моделирование / Т. Кватрани. – М. : ДМК Пресс, 2001.

44. Коберн, А. Быстрая разработка программного обеспечения / А. Коберн ; пер. с англ. – М. : ЛОРИ, 2002.

45. Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн ; пер. с англ. – М. : ЛОРИ, 2002.

46. Ковалев, И.А. Модели и алгоритмы синтеза кроссплатформенного контроллера автоматизации технологических процессов : диссертация на соискание ученой степени кандидата технических наук : 05.13.06 / И.А. Ковалев [Место защиты: Моск. гос. технол. ун-т «Станкин»].- Москва, 2017.- 157 с.: ил.

47. Ковалев, И.А. Модели и алгоритмы синтеза кроссплатформенного контроллера автоматизации технологических процессов: автореферат дис. … канд-та тех. наук: 05.13.06, / Ковалев Илья Александрович. - Москва, 2017. - 25 с

48. Козак, Н.В. Применение программно-реализованных логических контроллеров в системах автоматизации упаковочного оборудования / Н.В. Козак, Р.А. Нежметдинов // Автоматизация в промышленности. 2012. № 11. С. 23-28.

49. Конноли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика / Т. Конноли, К. Бегг ; пер. с англ. – 3-е изд. – М. : Вильямс, 2003.

50. Коновалов, Л.И. Элементы и системы электроавтоматики [Текст] : учеб. пособие для вузов по спец. «Автоматизация и комплексная механизация хим.-технол. процессов» / Л.И. Коновалов, Д. П. Петелин. - 2-е изд., перераб. и доп. - М. : Высшая школа, 1985. - 216 с.: ил., табл. : ил. - Библиогр.: с. 214.

51. Котляров, В.П. Основы тестирования программного обеспечения : Курс лекций / В.П. Котляров — Москва : Интуит НОУ, 2016. — 348 с. — ISBN 978-5-9556-0027-7. — URL: https://book.ru/book/917951 (дата обращения: 30.03.2024). — Текст : электронный.

52. Криспин, Л. Гибкое тестирование / Л. Криспин, Дж. Грегори. // Addison Wesley, 2010, 1st ed. – 466 с.

53. Кулаков, А.Ф. Управление качеством программных средств ЭВМ / А.Ф. Кулаков // Киев: Тэхника. – 1989 – С.169

54. Кулиев, А.У. Разработка способа аппаратно-независимого управления электроавтоматикой для сокращения времени выпуска различных компоновок токарно-фрезерных станков с ЧПУ – диссертация на соискание ученой степени

кандидата технических наук: 05.13.06 / Кулиев, Абай Уангалиевич [Место защиты: Моск. гос. технол. ун-т «Станкин»].- Москва, 2016.- 134 с.: ил. РГБ ОД, 61 09-5/2856

55. Куликов, С.С. Тестирование программного обеспечения Куликов. Базовый курс / С. С. Куликов. — Минск: Четыре четверти, 2017. — 312 с. ISBN 978-985-581-125-2

56. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман. – М. : Вильямс, 2001.

57. Леффингуэлл, Д. Принципы работы с требованиями к программному обеспечению. Унифицированный подход / Д. Леффингуэлл, Д. Уидриг ; пер. с англ. – М. : Вильямс, 2002.

58. Майерс, Гл. Искусство тестирования программ, 3-е издание. / Гл. Майерс, Т. Баджетт, К. Сандлер. – Москва: Диалектика, 2012. – 272 с.

59. Маклаков, С.В. Создание информационных систем с AllFusion Modeling Suite / С.В. Маклаков. – М. : Диалог-МИФИ, 2003.

60. Мартинов, Г.М. Автоматизация технологических процессов в машиностроении / Г.М. Мартинов, Л.И. Мартинова, Р.Л. Пушков // Учебное пособие - 2-е изд., перераб. и доп. - М.: МГТУ «Станкин», 2011. - 200 с.

61. Мартинов, Г.М. Автоматизация технологических процессов в машиностроении. Часть – I. Числовое программное управление / Г.М. Мартинов, Л.И. Мартинова, Р.Л. Пушков // Учебное пособие по подготовке специалистов с высшим профессиональным образованием для кадрового перевооружения машиностроительного комплекса России. М.: МГТУ СТАНКИН. 2010. 203 с.

62. Мартинов, Г.М. Декомпозиция и синтез программных компонентов электроавтоматики / Г.М. Мартинов, Н.В Козак. // Приборы и системы. Управление, контроль, диагностика. – 2006. – №12. – С. 4–11.

63. Мартинов, Г.М. Модульный подход к построению специализированной системы ЧПУ для обрабатывающих центров наклонной компоновки / Г.М. Мартинов, Р.А. Нежметдинов // СТИН. 2014. Т. 11. С. 28-33.

64. Мартинов, Г.М. Особенности реализации и специфика применения

функций многокоординатной обработки в системе ЧПУ «АксиОМА Контрол» / Г.М. Мартинов, А.И. Обухов, Р.Л. Пушков, С.В. Евстафиева // Автоматизация в промышленности, №5. 2017. с.17-22

65. Мартинов, Г.М. Подход к построению кроссплатформенного автономного контроллера автоматизации на базе синтеза его отдельных модулей / Г.М. Мартинов, А.С. Григорьев, И.А. Ковалев // Автоматизация в промышленности, №5. 2018. с.61-64

66. Мартинов, Г.М. Принципы построения кроссплатформенного программно реализованного контроллера электроавтоматики систем ЧПУ высокотехнологичными производственными комплексами / Г.М. Мартинов, Р.А. Нежметдинов, А.С. Емельянов // Вестник МГТУ Станкин. 2013. № 1 (24). С. 42-51.

67. Мартинов, Г.М. Развитие систем управления технологическими объектами и процессами / Г.М. Мартинов // Мир техники и технологий. – 2009. – № 6. – С. 34-35.

68. Мартинов, Г.М. Разработка средств визуализации и отладки управляющих программ для электроавтоматики, интегрированных в систему ЧПУ / Г.М. Мартинов, Р.А. Нежметдинов, П.А. Никишечкин // Вестник МГТУ Станкин. 2012. № 4 (23). С. 134-138.

69. Мартинов, Г.М. Системы числового программного управления для автоматизации технологических процессов машиностроительного комплекса России / Г.М. Мартинов, Л.И. Мартинова, Р.Л. Пушков // Учебное пособие - М.: МГТУ «Станкин», 2011. - 169 с.

70. Мартинов, Г.М. Современные машиностроительные технологии производства / Г.М. Мартинов, Ф.С. Сабиров, М.А. Волосова, Е.Д. Коршунова, Е.В. Лукашевич – М.:МГТУ «Станкин», 2011. - 159 с.

71. Мартинов, Г.М. Современные тенденции развития компьютерных систем управления технологического оборудования / Г.М. Мартинов // Вестник МГТУ «Станкин». – 2010. – №1. – С.119-125.

72. Мартинов, Г.М. Специфика построение панелей управления систем ЧПУ по типу универсальных программно-аппаратных компонентов /

Г.М. Мартинов, Н.В. Козак, Р.А. Нежметдинов // Автоматизация. Современные технологии. 2010. № 7. С. 34-40

73. Мартинов, Г.М. Специфика построения редактора управляющих программ электроавтоматики стандарта МЭК 61131 / Г.М. Мартинов, Р.А. Нежметдинов, П.А. Никишечкин // Вестник МГТУ Станкин. 2014. № 4 (31). С. 127-132.

74. Мартинов, Г.М. Специфика построения редактора управляющих программ электроавтоматики стандарта МЭК 61131 / Г.М. Мартинов, Р.А. Нежметдинов, П.А. Никишечкин // Вестник МГТУ «СТАНКИН». – 2014. – № 4 (31). – С. 127–132.

75. Мартинов, Г.М. Способ построения инструментария систем мониторинга и настройки параметров мехатронного технологического оборудования на основе специализированных программных средств / Г.М. Мартинов, Р.А. Нежметдинов, С.В. Соколов // Мехатроника, автоматизация, управление. 2012. № 7. С. 45-50.

76. Мартинов, Г.М. Формирование базовой вычислительной платформы ЧПУ для построения специализированных систем управления / Г.М. Мартинов, Л.И. Мартинова // Вестник МГТУ «Станкин», № 1 (24), 2014, с. 92-97.

77. Мартинов, Г.М. Цифровые производственные технологии согласно концепции Industry 4.0 / Г.М. Мартинов // Автоматизация в промышленности, №5. 2019. с.3-5

78. Мартинова, Л.И. Мировые тренды, возможности и перспективы развития систем ЧПУ станочного оборудования / Л.И. Мартинова, Г.М. Мартинов // СТИН, №7. 2019. с.28-31

79. Мегаобучалка. Над чем мы проводим тесты (модуль, группа модулей, система). [Электронный ресурс]. – Режим доступа: https://megaobuchalka.ru/9/5843.html (дата обращения: 12.03.2018).

80. Мещерякова, В.Б. Металлорежущие станки с ЧПУ / В.Б. Мещерякова, В.С. Стародубов // М.: Инфра-М, 2015. – 336 с.

81. Минаев, И.Г. Программируемые логические контроллеры [Текст]:

практическое руководство для начинающего инженера / И.Г. Минаев, В.В. Самойленко. – Ставрополь: АГРУС, 2009. – 100с. – ISBN 978-5-9596-0609-1.

82. Мишель, Ж. Программируемые контроллеры / Ж. Мишель, К. Лоржо, Б. Эспьо // Пер. с франц. А. П. Сизова. – М.: Машиностроение, 1986. – 176 с., ил.

83. Нежметдинов, Р.А. Повышение качества архитектурных решений систем ЧПУ на основе программно реализованного контроллера типа Soft PLS / Нежметдинов Р. А., Шемелин В. К. // Автоматизация. Современные технологии. 2008. № 6. С. 33-36.

84. Нежметдинов, Р.А. Повышение эффективности функционирования электроавтоматики станков с ЧПУ на основе реализации регулярных моделей архитектуры программно реализованных контроллеров типа SoftPLC : диссертация на соискание ученой степени кандидата технических наук : 05.13.06 / Нежметдинов Рамиль Амирович; [Место защиты: Моск. гос. технол. ун-т «Станкин»].- Москва, 2009.- 159 с.: ил. РГБ ОД, 61 09-5/2856

85. Нежметдинов, Р.А. Подход к построению систем логического управления технологическим оборудованием для реализации концепции «Индустрия 4.0» / Р.А. Нежметдинов, П.А. Никишечкин, И.А. Ковалев, Н.Ю. Червоннова // Автоматизация в промышленности. – 2017. – №5. – С.5-9.

86. Нежметдинов, Р.А. Построение специализированной системы ЧПУ для многокоординатных токарно-фрезерных обрабатывающих центров / Р.А. Нежметдинов, Р.Л. Пушков, С.В. Евстафиева, Л.И. Мартинова // Автоматизация в промышленности. 2014. № 6. С. 25-28.

87. Нежметдинов, Р.А. Принципы и методологические основы построения программных систем логического управления технологическим оборудованием / Р.А. Нежметдинов: диссертация на соискание ученой степени д-ра тех. наук: 05.13.06, // Москва, 2020. - 250 с.

88. Нежметдинов, Р.А. Принципы и методологические основы построения программных систем логического управления технологическим оборудованием: автореферат на соискание ученой степени д-ра тех. наук: 05.13.06, / Нежметдинов Рамиль Амирович. - Москва, 2020. - 43 с.

89. Нежметдинов, Р.А. Расширение функциональных возможностей систем ЧПУ для управления механо-лазерной обработкой / Р.А. Нежметдинов, С.В. Соколов, А.И. Обухов, А.С. Григорьев // Автоматизация в промышленности. 2011. № 5. С. 49-53.

90. Нежметдинов, Р.А. Управление электроавтоматикой токарных и токарно-фрезерных станков на базе Soft PLC / Р.А. Нежметдинов, А.У. Кулиев, А.Ю. Николушкин, Н.Ю. Червоннова, Автоматизация в промышленности. 2014. № 4. С. 49-51.

91. Новиков, С.О. Оптимизация управления электромеханической системой с переменными параметрами модифицированным принципом максимума / С.О. Новиков: диссертация на соискание ученой степени кандидата технических наук // Электронная библиотека диссертаций и авторефератов диссертаций Национальной библиотеки Беларуси. – 2011. – Белорусский национальный технический университет.

92. Олссон, Г. Цифровые системы автоматизации и управления / Г. Олссон, Дж. Пиани – СПб.: Невский Диалект, 2001. - 557 с.: ил.

93. Официальный сайт интернет портала «МирПром» [электронный ресурс]: офиц. сайт // интернет портал «МирПром». – Режим доступа: http://mirprom.ru/public/sovershenstvovanie-sistem-chpu.html (дата обращения 15.01.2017).

94. Официальный сайт компании «Балт-Систем» [электронный ресурс]: офиц. сайт // компания «Балт-Систем». – Режим доступа: https://www.bsystem.ru (дата обращения 03.07.2017).

95. Официальный сайт немецкого концерна TUV [электронный ресурс]: офиц. сайт // Промышленные предприятия. – Режим доступа: https://www.tuv.com/world/en/industrial-plants.html (дата обращения 31.07.2023).

96. Официальный сайт немецкого концерна TUV www.tuv.com [электронный ресурс]: офиц. Сайт // Сертификационная база данных TUV. – Режим доступа: https://www.certipedia.com/search/matching_system_certificates?locale=en&q= (дата

обращения 31.07.2023)

97. Официальный сайт Правительства РФ «Стратегия развития электронной промышленности Российской Федерации на период до 2030 года» [Электронный ресурс]. – М.: – 50 с. – Режим доступа: http://static.government.ru/media/files/1QkfNDghANiBUNBbXaFBM69Jxd48ePeY.pdf (дата обращения: 11.06.2021).

98. Официальный сайт тестирования программного обеспечения «Про Тестинг» http://www.protesting.ru/ (дата обращения 25.01.2017).

99. Официальный сайт фирмы «3S – Smart Software Solution» [электронный ресурс]: офиц. сайт // компания «3S – Smart Software Solution». – Режим доступа: http://www.codesys.com (дата обращения 03.10.2018)

100. Перл, И.А. Введение в методологию программной инженерии / И.А. Перл, О.В. Каленова // СПб.: Университет ИТМО, 2019. – 50 с. с ил.

101. Пероцкая, В.Н. Основы тестирования программного обеспечения : учеб. пособие / В. Н. Пероцкая, Д. А. Градусов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2017 – 100 с. ISBN 978-5-9984-0777-2

102. Петров, И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / И.В. Петров, под ред. проф, В.П. Дьяконова. — М.: СОЛОН-Пресс, 2004. — 256 с : ил. — (Серия «Библиотека инженера»)

103. Петрухин, В.А. Методы и средства программной инженерии / В.А. Петрухин, Е.М. Лаврищева. – URL: http://www.intuit.ru/department/se/swebok/12/2.html

104. Понятие программируемого логического контроллера (ПЛК). 22.02.2015. [Электронный ресурс]. – Режим доступа: https://studfile.net/preview/1862107/page:9/ (дата обращения: 12.03.2017).

105. Практические аспекты анализа лог-файлов технологического оборудования на примере набора данных “Mill Data Set” / Е.В. Путинцева, Р.А. Нежметдинов, И.А. Ковалев, Ш. Котырова, // Научный рецензируемый журнал. – М.: Вестник МГТУ «СТАНКИН», 2022. – №2(61) – С. 8-14.

106. Путинцева Е.В. Формализованный подход к описанию жизненных

циклов объектов технологического оборудования, оснащенного системами логического управления, и средств их тестирования // Научно-технический журнал. – М.: Промышленные АСУ и контроллеры, 2024. – №7 – С. 21-27.

107. Путинцева Е.В., Нежметдинов Р.А. Методика разработки систем логического управления технологическим оборудованием // Материалы VI международной научной интернет-конференция «Проблемы и перспективы развития научно-технологического пространства». Сборник докладов. - Вологда 2022. – С.573-579.

108. Путинцева Е.В., Нежметдинов Р.А. Подходы обнаружения ошибок в программном обеспечении систем логического управления технологическим оборудованием // Материалы XV всероссийской конференции с международным участием «Машиностроение: традиции и инновации (МТИ – 2022)». Сборник докладов. – М.: ФГБОУ ВО «МГТУ «СТАНКИН», 2022. – С. 135-143 // eLIBRARY ID: 47379716 // УДК: 002:621

109. Путинцева Е.В., Нежметдинов Р.А. Практические аспекты обнаружения ошибок при тестировании программ логического управления для ПЛК // Научный рецензируемый журнал. – М.: Вестник МГТУ «СТАНКИН», 2023. – №2(66) – С. 8-14.

110. Путинцева Е.В., Нежметдинов Р.А. Тестовые испытания систем логического управления технологическим оборудованием // Современная Российская наука: актуальные вопросы, достижения и инновации: сборник статей IV Всероссийской научно-практической конференции. Пенза: – МЦНС «Наука и Просвещение». – 2022. – С. 46-49. // ISBN 978-5-00173-329-4

111. Ручное и автоматизированное тестирование: как выбрать эффективный подход // Точка качества – сайт по услугам тестирования ПО и русскоязычных блогов на тему тестирования ПО. 02.02.2018. [Электронный ресурс]. – Режим доступа: https://tquality.ru/blog/ruchnoe-i-avtomatizirovannoe-testirovanie-kak-vybrat-ehffektivnyj-podkhod/ (дата обращения: 28.11.2020).

112. Свободная энциклопедия «Википедия» [электронный ресурс]: офиц.сайт // XSLT (eXtensible Stylesheet Language Transformations). – Режим

доступа: https://ru.wikipedia.org/wiki/XSLT (дата обращения 29.11.2018)

113. Селевцов, Л.И. Автоматизация технологических процессов: учебник для студ. учреждений сред. проф. образования / Л.И. Селевцов, А.Л. Селевцов. - 3-е изд., стер. – М.: Издательский центр «Академия», 2014. – 352 с.

114. Соломенцев, Ю.М. Автоматизация оценки надежности программного обеспечения для систем управления технологическими процессами / Ю.М. Соломенцев, С.А. Шептунов, Н.В. Суханова, И.С. Кабак // «Вестник БГТУ» № 3 (47) 2015

115. Сосонкин, В.Л. Программирование систем числового программного управления: Учеб. пособие. / В.Л. Сосонкин, Г.М. Мартинов – М. Логос, 2008. – 344 с. ISBN 978-5-98704-296-8.

116. Сосонкин, В.Л. Системы числового программного управления: Учеб. пособие. / В.Л. Сосонкин, Г.М. Мартинов – М. Логос, 2005. – 296 с. ISBN 5-98704-012-4.

117. Справочно-поисковая система // Интернет ресурс – URL: https://nntc-nnov.ru/2023/10/24/kak-postroit-ciklogrammu (дата обращения: 20.12.2023).

118. Тестирование программных продуктов // Рефератбанк – база данных рефератов различной тематики. 23.01.2002. [Электронный ресурс]. – Режим доступа: https://referatbank.ru/referat/preview/31516/referat-testirovanie-programmnyh-produktov.html (дата обращения: 27.10.2017).

119. Типовые нормативы времени на работы по подготовке и проведению экспериментов на испытательных стендах // М.: Институт труда, 2002. – 78 с.

120. Типовые нормы времени на программирование задач для ЭВМ. // М.: Экономика // Центральное бюро нормативов по труду государственного комитета по труду и социальным вопросам, 1988. – 143 с.

121. Типовые нормы времени на разработку технологической документации // М.: Экономика // Центральное бюро нормативов по труду государственного комитета по труду и социальным вопросам, 1991. – 143 с.

122. Тюрбеева, Т.Б. Разработка метода моделирования процессов жизненного цикла прикладных автоматизированных систем, обеспечивающего

формирование нормативно-методической среды их поддержки : автореферат дис. ... кандидата технических наук : 05.13.01 / Т.Б. Тюрбеева; [Место защиты: Моск. гос. технол. ун-т «Станкин»]. - Москва, 2015. - 27 с.

123. Укрупненные нормы времени на разработку программных средств вычислительной техники. / Межгосударственный совет по стандартизации, метрологии и сертификации. – М.: Издательство Экономика, 1988. – 65 с.

124. Уфимцева, Г. Перевод книги Ли Копланда “A Practitioner's Guide to Software Test Design” / Г. Уфимцева.

125. Фр. Брукс, Фр. Мифический человеко-месяц, или Как создаются программные системы, 2-е издание. / Фр. Брукс, Чапел Хилл. – Москва: Символ-Плюс, 2010. – 304 с.

126. Чеканин, В.А. Структура данных для задачи трехмерной ортогональной упаковки объектов / В.А. Чеканин, А.В. Чеканин // Вестник МГТУ «Станкин» № 1. 2015. С. 112-116.

127. Чеканин, В.А. Эвристический алгоритм оптимизации решений задачи прямоугольного раскроя / В.А. Чеканин, А.В. Чеканин // Вестник МГТУ «Станкин» № 4. 2014. С. 210-213.

128. Черушева, Т.В. Проектирование программного обеспечения / Т. В. Черушева // Пенза.: ПГУ, 2014. – 173 с.

129. Чуваков, А.Б. Основы подготовки и эффективной эксплуатации обрабатывающих станков с ЧПУ / А.Б. Чуваков, Д.В. Чиненков // Нижний Новгород: НГТУ, 2014. – 219 с.

130. Чуваков, А.Б. Технология изготовления деталей на станках с ЧПУ. Производственное оборудование и основы программирования операций / А.Б. Чуваков // Нижний Новгород, 2011 [Электронный ресурс]. – 280 с.

131. Шемелин, В.К. Применение технологии «клиент-сервер» при проектировании контроллера типа Soft PLC для решения логической задачи в рамках систем ЧПУ / В.К. Шемелин, Р.А. Нежметдинов // Автоматизация. Современные технологии. 2010. № 3. С. 20-24.

132. Шемелин, В.К. Применение технологии клиент-сервер при

проектировании контроллера типа SoftPLC для решения логической задачи в рамках систем ЧПУ / В.К. Шемелин, Р.А. Нежметдинов // Автоматизация и современные технологии. – 2010. – №3. – С. 31–37.

133. Шемелин, В.К. Совершенствование системы управления электроавтоматикой тяжелых станков на основе расширения средств динамического отображения состояний станков / В.К. Шемелин // Материалы Всероссийской молодежной конференции «Автоматизация и информационные технологии (АИТ-2011). Сборник докладов, том III. – М.: МГТУ «Станкин», 2011. – 173 с., с.64-70.

134. Шопырин, Д.Г. Управление проектами разработки ПО / Д.Г. Шопырин // Дисциплина «Гибкие технологии разработки программного обеспечения» [Электронный ресурс]. – СПб. : НИУ ИТМО, 2007 – 131 с. – Режим доступа: http://e.lanbook.com/book/43554.

135. Approach to Testing Logical Control Systems of Technological Equipment / Nezhmetdinov, R.A., Urinov, N.F., Derkach, E.V., Abdullaeva, D.H. // (2020) 2020 International Multi-Conference on Industrial Engineering and Modern Technologies, FarEastCon 2020, статья № 9271361

136. Copeland, L. A Practitioner's Guide to Software Test Design / Lee Copeland // Artech House. – 2004. – 355 с. // ISBN:158053791x

137. Jackvony, Kr. The Complete Software Tester / Kr. Jackvony // Concepts, Skills, and Strategies for High-Quality Testing. 2021 – 467 с.

138. Kaner, Cem. Lessons Learned in Software Testing / Cem Kaner, James Bach, Bret Pettichord – USA, Addison Wesley, 2001, 1st ed. – 316 с.

139. Kovalev, I. Approach to Assessing the Possibility of Functioning of CNC and PAC Systems on Various Software and Hardware Platforms / I. Kovalev, R.A. Nezhmetdinov, P.A. Nikishechkin // In: 2019 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon). Vladivostok: IEEE, pp.1-5. doi: 10.1109/FarEastCon.2019.8933999

140. Martinov, G. Approach in Implementing of Logical Task for Numerical Control on Basis of Concept “Industry 4.0” / G. Martinov, N. Kozak, R. Nezhmetdinov

// 2018 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM). ISBN**:** 978-1-5386-4307-5

141. Martinov, G. Trends in the numerical control of machine-tool systems / G. Martinov, L.I. Martinova // Russian Engineering Research. – 2010. T.30 - №10. – P. 1041-1045.

142. Martinova, L. An Approach to The Implementation of the Machine Safety Function Using an Integrated in the CNC System Softplc and an External Safety Controller Made According to the SoftPlC / L. Martinova, G. Martinov, 2022 International Russian Automation Conference (RusAutoCon), 2022, pp. 961-965, doi: 10.1109/RusAutoCon54946.2022.9896367

143. Mathematical criteria for testing the logical control programs for technological equipment / Ramil Nezhmetdinov, Elena Putintseva, Dilnavoz Abdullaeva, and Dusmurod Bafoev // E3S Web of Conferences 390, 03006 (2023) – Agritech-VIII 2023 – https://doi.org/10.1051/e3sconf/202339003006

144. Nezhmetdinov, R. An approach to the development of logical control systems for technological equipment in the concept of Industry 4.0 / R. Nezhmetdinov, I. Kovalev, N. Chervonnova, R. Nezhmetdinova, A. Al Khoury // ICMTMTE 2020, MATEC Web of Conferences 329, 03044 (2020). doi: 10.1051/matecconf/202032903044

145. Nezhmetdinov, R.A. Extending the functional capabilities of NC systems for control over mechano-laser processing / R.A. Nezhmetdinov, S.V. Sokolov, A.I. Obukhov, A. S. Grigor'Ev // Automation and Remote Control. 2014. T. 75. № 5. C. 945-952.

146. Nikishechkin, P. Practical aspects of constructing a specialized control system for compact machines / P. Nikishechkin, I. Kovalev, O. Kazaryan, N. Grigorev // ICMTMTE 2020, IOP Conf. Series: Materials Science and Engineering 971 (2020) 042085. doi: 10.1088/1757-899X/971/4/042085

**ПРИЛОЖЕНИЕ А – Справки об использовании результатов диссертационного исследования**

ОБЩЕСТВО С ОГРАНИЧЕННОЙ ОТВЕТСТВЕННОСТЬЮ
**СТАНКОТЕХНИКА**

THE LIMITED LIABILITY COMPANY
**STANKOTEKHNIKA**

Россия, 300002, г. Тула, ул. Мосина, 2 — 2 Mosin Str., Tula, 300002, Russia
☎ +7(4872)32-11-57 • Fax: +7(4872)36-96-22 • E-Mail: stanki95@bk.ru

УТВЕРЖДАЮ
Директор
ООО «Станкотехника»
Шелыгова С.А.
«08» 10 2024

**СПРАВКА**
об использовании результатов диссертационного исследования
Путинцевой Елены Валентиновны

Результаты диссертационного исследования Путинцевой Елены Валентиновны на тему «Модели и алгоритмы тестирования систем логического управления с использованием специализированных испытательных стендов» были использованы ООО «Станкотехника» на токарно-фрезерных обрабатывающих центрах ВТМ-250 собственного производства в виде:

1) разработки программы логического управления на языке функциональных блоков, а также применения алгоритмов тестирования, предложенных в ходе выполнения диссертационной работы для проверки работоспособности автоматизированных систем станка ВТМ-250;

2) применения методики тестирования систем логического управления, основанной на разработке и применении испытательного стенда, для проведения комплексной проверки корректности работы технологического оборудования без риска поломки дорогостоящих узлов и деталей.

Использование результатов диссертационного исследования в практической деятельности позволяет сократить время тестирования и отладки систем промышленной автоматики.

Заместитель директора
по станкостроительному производству (подпись) Трунов С. Н.

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

Вадковский пер., д. 1, Москва, ГСП-4, 127994. Тел.: (499) 973-30-76. Факс: (499) 973-38-85
E-mail: rector@stankin.ru

**СПРАВКА**

**об использовании в учебном процессе**

**результатов диссертационного исследования**

**Путинцевой Елены Валентиновны**

Результаты диссертационного исследования Путинцевой Елены Валентиновны на тему «Модели и алгоритмы тестирования систем логического управления с использованием специализированных испытательных стендов» использовались в виде программ и методических материалов в учебном процессе кафедры компьютерных систем управления ФГБОУ ВО «МГТУ «СТАНКИН» в рамках освоения дисциплин:

- «Автоматика и управление движением», «Автоматизация технологических процессов и производств» подготовки бакалавров по направлению 15.03.04 «Автоматизация технологических процессов и производств»;
- «Программируемые логические контроллеры в системах управления», «Проектирование систем автоматизации и управления» подготовки магистров по направлению 15.04.04 «Автоматизация технологических процессов и производств».

Проректор по образовательной
деятельности и молодежной политике
ФГБОУ ВО «МГТУ «СТАНКИН»,
канд. техн. наук, доцент _____ /Бильчук Мария Викторовна/

«18» сентября 2024