

Утверждаю
генеральный директор
ООО «3В Сервис»



Петухов В.Н.



Среда динамического моделирования технических систем SimInTech™

План верификации

Модуль генерации кода для систем реального времени

ШИФР ГК16ПВ

Москва, 2016



Аннотация

В данном программном документе приведено описание мероприятий по верификации программного обеспечения, на предмет соответствия техническому заданию.

Указаны основные процедуры и этапы верификации, а также набор документов для фиксации результатов верификации.

Данный документ может быть использован для независимой верификации модуля генерации кода систем реального времени ПО SimInTech, в том числе силами заказчика.



СОДЕРЖАНИЕ

Аннотация	2
1. Введение	4
2. Основание для разработки	6
3. Термины и определения	7
4. Цели и задачи процесса верификации	14
4.1 Стратегия верификации	16
4.1.1 Верификация тестовой версии	16
4.1.2 Верификация обычной (актуальной) версии	17
5. Требования к аппаратному и программному обеспечению	18
5.1 Требования к аппаратному обеспечению	18
5.2 Требования к программному обеспечению	19
5.3 Требования к тестовым примерам	19
6. Этапы процесса верификации ПО	20
6.1 Сравнение результатов генерации кода старой и новой версии ПО	22
6.1.1 Последовательность тестирования	22
6.1.2Arteфакты этапа верификации	25
6.2 Верификация создания кода для новых блоков	26
6.2.1 Модульное тестирование блоков	27
6.2.1.1 Требования к тестовому проекту	27
6.2.1.2 Последовательность тестирования	31
6.2.2Arteфакты этапа верификации	32
6.3 Тестирование блоков и функций в прикладной программе	33
6.3.1 Требования к тестовому окружению	33
6.3.2 Последовательность тестирования	40
6.3.3 Arteфакты этапа верификации	41
ПРИЛОЖЕНИЕ 1. Текст блока анализа результатов тестирования	42
ПРИЛОЖЕНИЕ 2. Примеры таблиц результатов	44



1. Введение

План верификации составлен для модуля генерации кода систем реального времени, входящего в состав SimInTech.

Программа для ЭВМ “Среда динамического моделирования SimInTech” (сокращенное название «SimInTech»), свидетельство о регистрации №2010617758 - современная среда интеллектуальной системы автоматизированного **проектирования** (САПР), предназначенная для детального исследования и анализа нестационарных процессов в системах автоматического управления, в следящих приводах и роботах, в любых технических системах, описание динамики которых может быть реализовано методами структурного моделирования.

SimInTech обеспечивает создание алгоритмов управления в виде функционально-блочных диаграмм. ПО содержит в себе математическое ядро для проведения динамического расчета созданного алгоритма управления путем задания входных воздействий и анализа изменений внутренних параметров и выходных значений во время моделирования.

SimInTech является базовым программным обеспечением для верифицируемого модуля. Комплексная система моделирования систем управления и программирования приборов включает в себя:

- модуль генерации кода для автоматической генерации исходных кодов и исполняемых модулей;
- среду разработки для проектирования алгоритмов управления в виде наглядных функционально-блочных диаграмм;
- систему исполнения программ для компьютеров систем управления, для выполнения сгенерированных при помощи генератора кода исполняемых модулей.

Под компьютером (прибором) следует понимать программируемые логические контроллеры (ПЛК) в составе промышленных компьютеров, работающие под управлением POSIX-совместимых операционных систем реального времени.

Предметом данного плана является верификация модуля генерации кода Си, на предмет его соответствия техническому заданию, а также требованиям ГОСТ Р МЭК 60880 – 2011



«Атомные станции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А».

Верификация должна проводиться персоналом, не задействованным в разработке ПО.



2. Основание для разработки

Основанием для разработки является:

Договор N 437 - 01 от 26.07.2013 по теме: «Разработка программного обеспечения верхнего уровня программно-технического комплекса средств автоматического управления».

Заказчик ООО «Московский завод «ФИЗПРИБОР».

SimInTech. Техническое задание. Модуль генерации кода систем реального времени.

SimInTech. Руководство пользователя.

План особо важных работ по доработке программного обеспечения на 2015 год. Утвержден 15.02.2013. ООО «ЗВ Сервис».

Требования к плану верификации программного обеспечения со стороны следующих стандартов:

- ГОСТ Р МЭК 61513-2011;
- ГОСТ Р МЭК 62138-2010;
- ГОСТ Р МЭК 60880-2011.



3. Термины и определения

3.1 Анимация (animation): процесс, посредством которого указанное в спецификации поведение демонстрируется с реальными значениями, полученными из задающих поведение выражений и некоторых входных величин.

3.2 Алгоритм (algorithm): набор последовательности работы ПО для преобразования входных данных программы или подпрограммы в выходные данные.

3.3 Схема алгоритма (algorithm diagram): алгоритм, составленный в виде графической схемы (функционально-блочной диаграммы) на проблемно ориентированном языке SimInTech.

3.4 Лист алгоритма (algorithm page): схема алгоритма в SimInTech, оформленная в соответствии с требованиями к оформлению спецификации ПО - графическое изображение алгоритма управления или его части в виде функционально-блочной диаграммы. Является частным случаем расчетной схемы SimInTech.

3.5 Группа алгоритмов (group of algorithm): совокупность листов алгоритмов, объединённые в субмодель в среде SimInTech.

3.6 Расчетная схема SimInTech (SimInTech simulation diagram): структурная схема, созданная в окне графического редактора SimInTech, описывающая на предметно-ориентированном языке математическую модель алгоритма, процесса или объекта, динамику поведения которого во времени, можно представить в виде системы алгебраических и дифференциальных уравнений в форме Коши. На основании расчетной схемы ядро SimInTech обеспечивает математическое моделирование динамического поведения объекта во времени с заданной точностью.

3.7 Блок (block): базовый элемент расчетной схемы (функционально-блочной диаграммы в SimInTech). Основными атрибутами блока являются его графический образ, свойства (задаваемые пользователем), параметры (вычисляемые блоком), математическая модель, входные и выходные порты. Графическое изображение может быть статическим или анимированным.

В математическом аспекте блоки представляют собой операторы преобразования входных сигналов блока в его выходные сигналы. Совокупность блоков и соединяющих их порты



линий связи образует расчетную схему объекта или алгоритм прикладного программного обеспечения в SimInTech.

3.8 Библиотека блоков (block library): набор блоков, объединенных по определенному (в основном, функциональному) признаку. Набор блоков содержится в файле с расширением «csl» и предназначен для набора проекта SimInTech.

3.9 Видеокادر (mnemo): проект SimInTech в виде интерактивной и анимированной структурной схемы, позволяющий при моделировании оказывать воздействие на алгоритм или модель и наблюдать результаты работы.

3.10 Проект SimInTech (SimInTech project): файл, содержащий расчетную схему, созданную в графическом редакторе SimInTech, сохраненный на диске в виде бинарного и/или текстового файла с уникальным именем и расширением «prt» (для бинарного) и «xprt» (для текстового) файла. Проект SimInTech содержит расчетную схему – математическую модель, предназначенную для расчета тем или иным математическим решателем или расчетным кодом.

3.11 Пакет SimInTech (SimInTech pack): - файл, содержащий перечень проектов SimInTech и порядок их совместного запуска на расчет (моделирование), имеющий расширение «pak» и являющийся основным файлом для организации комплексной модели. Проекты, запускаемые на расчет в пакетном режиме, имеют одну базу сигналов в памяти компьютера и единый синхронизатор расчетного (модельного) времени, за счет чего они могут обмениваться значениями граничных (входных и выходных) сигналов между собой на каждом шаге расчета и осуществлять моделирование в едином синхронном модельном времени.

3.12. Прикладная функция (application function): функция системы контроля и управления, выполняющая задачу, связанную с контролируемым процессом, а не с функционированием самой системы.

3.13. Проблемно-ориентированный язык (application oriented language): компьютерный язык, специально разработанный для определенного типа применений и используемый лицами, являющимися специалистами в данном типе применений. В рамках данного документа графическое представление алгоритмов в виде функционально-блочной диаграммы SimInTech рассматривается как вид проблемно-ориентированного языка.



3.14. Прикладное программное обеспечение (application software): часть программного обеспечения системы контроля и управления, которая обеспечивает выполнение прикладных функций.

3.15. Автоматизированная генерация кода (automated code generation): функция автоматизированных инструментов, позволяющая преобразовывать проблемно-ориентированный язык в форму, пригодную для компиляции и выполнения на целевой системе.

3.16. Идентификатор объекта (identification): целое число (тип данных integer), являющееся уникальным внутренним именем каждого объекта на схеме и используемое для прямого обращения к объекту на низком уровне во многих функциях встроенного языка программирования.

3.17. Комплексная модель (complex model): совокупность проектной базы сигналов и файлов проектов, содержащих расчетные схемы алгоритмов прикладной программы и математическую модель объекта управления, созданных в SimInTech. Комплексная модель является виртуальным объектом, динамическое поведение которого совпадает с поведением реального технического объекта с заданной степенью точности.

3.18. Линия связи (connection line): служебный блок в виде полилинии, второй базовый элемент расчетной схемы, соединяющий выходной порт одного блока и входной порт другого блока. В общем случае линия связи может соединять множество входных портов с одним выходным портом. В математическом аспекте линии связи являются шинами данных (сигналов) и осуществляют направленную передачу данных от выходов блока к входам других блоков.

3.19. Параметр блока (blocks parameter): формируемая (вычисляемая) блоком переменная, характеризующая работу блока.

3.20. Свойство блока (blocks property): задаваемая пользователем характеристика (константа или переменная величина, какого-либо определённого типа данных) для работы блока.

3.21. Стандартная подпрограмма (standard program): см. типовое решение.

3.22. Субмодель (макроблок) (sub model): блок, математическая модель которого задана в виде структурной схемы, расположенной «внутри» субмодели. Макроблоки позволяют реализовать принцип вложенности структурных схем, являются одним из механизмов



создания и сохранения в библиотеке новых типов блоков. Математическая модель макроблока может быть сохранена в отдельном файле (в отдельном проекте) для многократного использования в других проектах.

3.23. Типовое решение (standard program): один или несколько связанных специальных алгоритмов, определяющих принцип управления, и (или) измерения данного типа объекта. Применяется для генерации типового алгоритма для объектов данного типа. Например, типовым решением может быть способ контроля достоверности датчика (на обрыв, зашкал, непревышение скорости изменения измеряемой величины, перевод измеренного сигнала в физические единицы измерения для дальнейшего использования в алгоритмах). Типовое решение также называют стандартной подпрограммой.

3.24. Типовой алгоритм (standard algorithm): алгоритм конкретного объекта (устройство управления, канал измерения), созданный автоматически на основе типового решения для данного объекта.

3.25. Канал (channel): совокупность взаимосвязанных компонентов внутри системы, имеющая один выход. Канал теряет свою идентичность тогда, когда сигналы на единственном выходе сочетаются с сигналами от других каналов, например, от канала контроля или от канала активизации защиты. [Глоссарий МАГАТЭ NS-G-1.3]

3.26. Вычислительный прибор (вычислительный узел, computing unit) (см. Компьютер).

3.27. Компьютер (computer): программируемое функциональное устройство, которое состоит из одного или нескольких процессоров и периферийного оборудования, управляется хранящимися внутри программами и способно выполнять основные вычисления, включая многочисленные арифметические или логические операции без вмешательства в этот процесс человека.

Примечание – Компьютер может быть автономным устройством или может состоять из нескольких взаимосвязанных устройств.

3.28. Компьютерная программа (computer program): набор упорядоченных команд и данных, которые описывают операции в форме, приемлемой для их выполнения компьютером.

3.29. Компьютерная система (computer-based system): система контроля и управления, функции которой, в большей своей части, зависят от использования микропроцессоров,



программируемого электронного оборудования или компьютеров, либо полностью определяются таким использованием.

Примечание – Эквивалентно следующему: цифровые системы, системы с программным обеспечением, программируемые системы.

3.30. Данные (data): представление информации или команд в виде, пригодном для передачи, интерпретации или обработки с помощью компьютера.

Примечание – Данные, необходимые для определения параметров и для реализации прикладных и служебных функций в системе называются «прикладными данными».

3.31. Динамический анализ (dynamic analysis): процесс оценки системы или компоненты, основанный на их поведении в процессе работы. В противоположность статическому анализу. [IEEE 610]

3.32. Отказ (failure): отклонение реальной работы от запланированной.

3.33. Универсальный язык (general-purpose language): компьютерный язык, предназначенный для всех видов применения.

Примечание 1 – Программное обеспечение операционной системы групп оборудования обычно реализуется с использованием универсальных языков.

Примечание 2 – Примеры: Ада, Си, Паскаль.

3.34. Инициализировать (initialize): установить счетчики, переключатели, адреса или содержимое устройств памяти на нулевое значение или на другие начальные величины в начале или в заданной точке выполнения компьютерной программы.

Для математической модели установить все переменные состояния в начальное положение, задать значения всех переменных, необходимых для проведения расчета.

3.35. Комплексные тестирования (integration tests): тестирования, проводимые во время процесса интеграции технического и программного обеспечения до валидации компьютерной системы с целью проверки совместимости программного обеспечения и технического обеспечения компьютера.

3.36. Библиотека (library): набор связанных элементов ПО, сгруппированных вместе, но индивидуально отбираемых для включения в окончательный продукт ПО.

3.37. Операционное системное программное обеспечение (operation system software): программное обеспечение, выполняемое на целевом процессоре во время работы, такое как драйверы и сервисы ввода/вывода, управление прерываниями, планировщик, драйверы



связи, библиотеки прикладных программ, диагностирование во время работы, управление резервированием и смягченной деградацией.

3.38. Ролевое управление доступом (role-based access control): управление доступом на основе правил, определяющих разрешение доступа пользователей к объекту (функции, данные) не на индивидуальном основании, а на основании принадлежности к группам с идентичными задачами.

3.39. Сигнальная траектория (signal trajectory): динамика изменения всех состояний оборудования, внутренних состояний, входных сигналов и действий оператора, определяющих выходы системы.

3.40. Программное обеспечение (ПО) (software): программы (т.е. набор упорядоченных команд), данные, правила и любая соответствующая документация, относящаяся к работе компьютерной системы контроля и управления. [МЭК 62138, 3.27]

3.41. Разработка ПО (software development): стадия жизненного цикла ПО, которая приводит к созданию ПО системы контроля и управления или программного продукта. Она охватывает деятельность, начиная от спецификации требований и до валидации и установки на объекте.

3.42. Модификация ПО (software modification): изменение в уже согласованном документе (или документах), ведущее к изменению рабочей программы.

Примечание – Модификации ПО могут происходить либо в процессе первоначальной разработки ПО (например, устранение ошибок, обнаруженных на поздних этапах разработки), либо когда ПО уже находится в эксплуатации.

3.43. Версия ПО (software version): экземпляр программного продукта, полученный путем модификации или корректировки предыдущего программного продукта.

3.44. Спецификация (specification): документ, в котором полным, точным и проверяемым образом изложены требования, проектные свойства и другие характеристики системы или компоненты и, часто, процедуры подтверждения удовлетворения этим требованиям.

Примечание – Существуют различные типы спецификаций, например, спецификация требований к ПО или спецификация проекта.

3.45. Статический анализ (static analysis): процесс оценки системы или ее компоненты, основанный на ее форме, структуре, содержании или документации. В дополнение к динамическому анализу.



3.46. Системное ПО (system software): часть ПО системы контроля и управления, созданная для конкретного компьютера или семейства оборудования с целью облегчения разработки, эксплуатации и модификации этих объектов и связанных с ними программ.

3.47. Валидация системы (system validation): подтверждение путем проверки и предоставления других свидетельств того, что система в целом соответствует спецификации требований (функциональность, время отклика, устойчивость к дефектам и ошибкам, запас прочности).

3.48. Верификация (verification): подтверждение путем проверки и предоставления объективных свидетельств того, что результаты деятельности соответствуют целям и требованиям, определенным для этой деятельности.



4. Цели и задачи процесса верификации

Основным назначением процесса верификации является доказательство корректности работы модуля генерации кода в составе SimInTech и его соответствия требованиям технического задания и требованиям ГОСТ Р МЭК 60880 – 2011 «Атомные станции. Системы контроля и управления, важные для безопасности. Программное обеспечение компьютерных систем, выполняющих функции категории А».

В процессе верификации должны быть выполнены проверки следующих фактов:

- Модуль генерации кода создает код, функционал которого соответствует функционалу исходной математической модели в SimInTech (рис. 1).
- Проверка того, что модуль генерации кода SimInTech соответствует требованиям технического задания и ГОСТ Р 60880 – 2011.
- Проверка того, что модуль генерации кода в составе SimInTech может быть использован для реализации типового процесса создания прикладного программного обеспечения для систем контроля и управления, важных для безопасности АЭС.
- Проверка того, что в ситуациях, не отраженных в требованиях, программное обеспечение ведет себя адекватно, то есть не происходит отказ системы.
- Проверка программного обеспечения на предмет типичных ошибок, которые делают программисты.

Процесс верификации выполняется для модуля генерации кода, проводится для новой версии. Предыдущая версия модуля генерации кода SimInTech имеет сертификат соответствия № РОСС RU.0001.01АЭ00.77.10.2446.

В процессе верификации нужно доказать, что код прикладного программного обеспечения, создаваемый новой версией модуля генерации кода, соответствует коду, создаваемому предыдущей верифицированной версией кода.

Типовой процесс разработки прикладного ПО подразумевает использование модуля генерации для преобразования прикладного ПО, созданного в виде программы на графическом предметно-ориентированном языке SimInTech в исходный код на



универсальном языке программирования, пригодном для создания программ для различных программно-аппаратных платформ.

Процесс верификации, описанный в данном документе, направлен на проверку того, что исходный код, созданный модулем генерации кода, обеспечивает создание прикладного ПО системы управления, работа которого соответствует работе модели, рассчитываемой расчетным ядром SimInTech (рис. 1).



Рисунок 1. Требования к эквивалентности модели и сгенерированного кода

В рамках данного документа рассматривается только верификация процесса создания исходного кода на универсальном языке программирования (синяя часть схемы).



4.1 Стратегия верификации

Каждое изменение в ПО SimInTech проходит внутреннюю проверку силами разработчика, путем тестирования на предмет соответствия изменений поставленным целям при разработке. Данное внутреннее тестирование осуществляется в режиме «белого ящика» и осуществляется непосредственным разработчиком.

Кроме внутреннего тестирования существует независимый процесс верификации ПО.

Верификация ПО осуществляется в режиме «черного ящика» специально созданной командой специалистов, не участвующих в разработке.

В рамках разработки ПО SimInTech предусмотрено два варианта выпуска версий:

- Тестовая версия. Выпускается командой разработчиков по завершению внутреннего тестирования силами разработчиков. Содержит исправления и доработки, созданные в рамках плановой деятельности на основании задач, сформированных руководством в системе Redmine.
- Обычная (актуальная) версия. Версия, прошедшая процедуру верификации сотрудниками, не участвующим в процессе внесения изменений и специалистами по верификации.

4.1.1 Верификация тестовой версии

Тестовая версия создается разработчиком на локальном рабочем месте и не передается другим участникам разработки, до завершения предварительного тестирования.

Методы тестирования, изложенные в данном документе, применяются разработчиками для систематического тестирования тестовых версий ПО перед их выпуском. Объем тестирования определяется самим разработчиком и определяется разработчиком самостоятельно на основании изменений, внесенных в тестовую версию.

Изменения в тестовой версии создаются на основании задач, вынесенных в систему управления разработкой Redmine.

Тестированию разработчиком в первую очередь должны быть подвергнуты выполнение требований, описанных к разработчику в виде задачи в Redmine.

В зависимости от содержания изменений в ПО тестированию могут подвергаться различные части системы. Набор тестов определяется разработчиком исходя из задач, решенных в рамках доработки и изменениях в ПО SimInTech.



Обязательная верификация тестовой версии разработчиком с использованием методов, описанных в данном документе, осуществляется в следующих случаях:

- **Изменение в генераторе кода.** Осуществляется проверка соответствия сгенерированного кода новой версии и последней верифицированной версии.
- **Изменения в ПО SimInTech, связанные с интерфейсом подготовки данных.** В этом случае тестируется возможность работы нового интерфейса с существующими проектами, созданными в предыдущей версии. Проверяется, что старые файлы проектов могут быть открыты в новой версии ПО.

В случае если тестирование, проведённое разработчиком показывает наличие ошибок в ПО, выпуск тестовой версии не осуществляется и разработчик выполняет необходимую работу по исправлению обнаруженных ошибок.

В случае, когда тестирование разработчиком не показывает наличие ошибок, версия выпускается как тестовая и становится доступной для установки заказчиками и группой тестирования.

В информацию по новой версии вносятся описания задач из Redmine, решенных программистами в результате выпуска данного обновления.

Данная запись свидетельствует о выполнении разработчиком внутреннего тестирования внесенных изменений.

4.1.2 Верификация обычной (актуальной) версии

Обычная версия выпускается при накоплении значительного количества изменений и (или) по требованию заказчика, при завершении работ по техническому заданию.

Верификация обычной (актуальной) версии в части модуля генерации кода осуществляется согласно плану верификации, представленному в данном документе.

Для верификации создается комиссия, включающая в себя сотрудников не участвующих в разработке модуля генерации кода.

Тестовая версия, прошедшая верификацию, становится актуальной версией ПО.



5. Требования к аппаратному и программному обеспечению

Для верификации модуля генерации кода должно быть использовано рабочее место, оснащенное компьютером, параметры которого соответствуют конфигурации, рекомендованной для пользователей ПО «SimInTech».

Специальные требования к конкретной системе не устанавливаются. Тестирование может проходить на любой версии Windows, по усмотрению ответственного за тестирование.

Требования к наличию или отсутствию системного, служебного и прикладного программного обеспечения (антивирусы, фаерволы, офисные программы, средства разработки, и т.п.) не устанавливаются.

В случае конфликта между SimInTech и модуля генерации кода и каким-либо ПО, установленным на рабочей станции тестирования, их необходимо фиксировать и сообщать разработчику.

Тестирование проводить только после устранения явных конфликтов между SimInTech и установленным программным обеспечением.

Перед началом процесса верификации описание конкретной конфигурации используемой программно-аппаратной платформы должно быть включено в отчет по верификации.

5.1 Требования к аппаратному обеспечению

- Процессор Intel или AMD с частотой не менее 1500 МГц
- Оперативная память не менее 4 Гб
- Жесткий диск не менее 500 Гб
- Монитор с разрешением 1920x1280 или больше.
- Параметры видеокарты в соответствии с требованиями используемой операционной системы Windows.



5.2 Требования к программному обеспечению

- Операционная система Windows 7 или старше
- Тестируемая версия ПО «SimInTech».
- ПО для сравнения текстовых файлов WinMerge Version 2.14.0.0 или другие средства сравнения текстовых файлов (например, Total Commander).
- Кросс-платформенные средства для компиляции исполняемых модулей MinGW 8.3 или старше.
- Редактор подготовки отчетов в формате MS Word.

5.3 Требования к тестовым примерам

В качестве тестовых примеров должны быть использованы:

- 1) Эталонный набор тестовых проектов SimInTech. Эталонные проекты содержат блоки, поддерживаемые предыдущей верифицированной версией модуля генерации кода SimInTech.
- 2) Набор исходных кодов, сгенерированных предыдущей версией SimInTech из набора тестовых проектов SimInTech.
- 3) Набор тестовых проектов для всех блоков и функций языка программирования, генерация которых поддерживается новой версией SimInTech.



6. Этапы процесса верификации ПО

Для разработки модуля генерации кода используется итерационный процесс, на каждой итерации используются каскадный жизненный цикл. В качестве основы для разработки взят существующий модуль генерации кода, в рамках работ осуществляется расширение его функциональности за счет добавления возможности генерации кода на универсальном языке программирования с использованием расширенного набора блоков функционально-блочной диаграммы, а также функций и операторов языка программирования.

Структура изменения функциональности модуля генерации кода в процессе разработки представлена на рисунке 2.

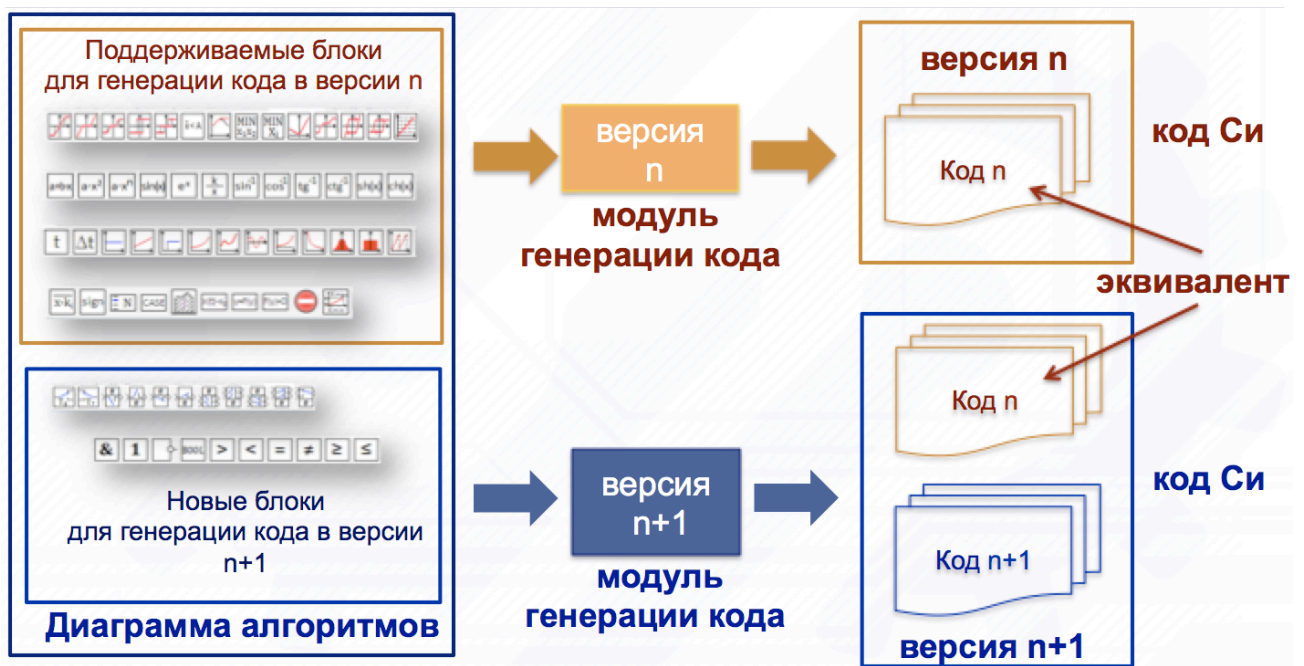


Рисунок 2. Изменение модуля генерации кода n процессе разработки.

В результате создания каждой новой версии модуля генерации кода, необходимо обеспечить следующее:

Совместимость диаграммы алгоритмов, используемых для предыдущего модуля генерации кода и нового модуля генерации кода (п. 5.5.2 Технического задания на модуль генерации кода).



Исходный код, созданный новой версией модуля генерации кода для предыдущей версии диаграмм алгоритмов, должен быть эквивалентен исходному коду, создаваемому верифицируемой версией (п. 5.1.3.2 Технического задания на модуль генерации кода).

Данный план верификации используется для проведения верификация после каждого изменения модуля генерации и создания новой версии ПО SimInTech.

Верификация осуществляется в 3 этапа:

1. Сравнение кода, сгенерированного предыдущей версией модуля генерации кода и новой версией.
2. Тест генерации кода для новых блоков и функций языка программирования, включенных в тестируемую версию генератора кода.
3. Сравнение работы модели в SimInTech и прикладного ПО, созданного из исходных кодов, сгенерированных модулем генерации кода.



6.1 Сравнение результатов генерации кода старой и новой версии ПО

Цель данного этапа верификации – доказать, что изменения внесенные в модуль генерации кода SimInTech не внесли ошибок в существующий функционал предыдущей версии. На данном этапе осуществляется проверка:

Совместимости файлов проектов сертифицированной версии SimInTech и тестируемой версии модуля генерации кода (п. 5.5.2 Технического задания на модуль генерации кода).

Эквивалентность результатов сертифицированной версии генератора кода и тестируемой версии (п. 5.1.3.2 Технического задания на модуль генерации кода).

Данное тестирование производится с использованием последнего дистрибутива ПО «Среда динамического моделирования технических систем «SimInTech» (ПО SimInTech), в который входит тестируемая версия модуля генерации кода.

Исходные данные для тестирования:

- Набор тестовых проектов SimInTech. Данный набор содержит проекты, в которых присутствуют все блоки, для которых предусмотрена генерация кода, актуальный для предыдущей сертифицированной версии модуля генерации кода SimInTech. При установке по умолчанию набор тестовых файлов находится в директории. *SimInTech/Demo/Тесты генерации кода блоков*.
- Набор текстовых исходных кодов, созданный в предыдущей верифицированной версии модуля генерации кода. Данный набор исходных кодов сохраняется для каждой референтной версии модуля генерации кода SimInTech.

6.1.1 Последовательность тестирования

- Используя референтную версию SimInTech, создать конфигурацию для массовой генерации кода из файлов.
- Добавить файлы из тестового набора в список файлов для генерации кода.
- Настроить директорию сохранения кодов.
- Осуществить генерацию кода для всех тестовых файлов в референтной версии SimInTech.



- Используя тестируемую версию SimInTech, создать конфигурацию для массовой генерации кода из файлов проектов, на основе конфигурации созданной для рефрентной версии.
- Добавить файлы из тестового набора в список файлов для генерации кода.
- Настроить директорию сохранения кодов.
- Осуществить генерацию кода для всех тестовых файлов в новой версии.
- Используя ПО сравнения текстовых файлов осуществить сравнение кода, а именно: сравнение всех файлов, полученных в результате генерации с тестовыми кодами.

Возможные результаты, выводы и дальнейшие действия представлены в таблице 1

Таблица 1. Результаты первого этапа верификации.

	Полученные результаты	Выводы	Дальнейшие действия
1	В процессе открытия тестовых файлов произошли ошибки и файлы исходных кодов не созданы или созданы не в полном объеме.	Верификация не пройдена. Ошибка совместимости в ПО «SimInTech»	Отчет об ошибках передается разработчикам
2	Файлы исходных кодов созданы в полном объеме. Полученные исходные коды не отличаются от тестовых образцов.	Этап верификация пройден.	Переход к следующему этапу верификации.
3	Файлы исходных кодов созданы в полном объеме. Полученные исходные коды отличаются от тестовых образцов.	Требуется дополнительная проверка отклонений	Отличия фиксируются и производится анализ отклонений.
4	Отклонения в файлах исходных кодов приводят к изменению функционала прикладного ПО.	Верификация не пройдена.	Отчет об ошибках передается разработчикам
5	Отклонения в исходных кодах не	Этап верификации	Блоки для которых в



	приводят к изменению функционала прикладного ПО.	пройден.	файлах исходных кодов обнаружены изменения, проходят дополнительную верификацию вместе с новыми блоками.
--	--	----------	--



6.1.2 Артефакты этапа верификации

По результатам данного этапа создается папка, содержащая файлы с исходными кодами, созданными верифицированной версией генератора кода.

А также выпускается отчет, содержащий:

- 1) Описание аппаратной и программной части рабочего места тестировщика.
- 2) Схемы тестовых моделей.
- 4) Отчет работы программы сравнения текстовых файлов.
- 5) Обоснование отклонений в текстах исходного кода.
- 6) Таблица результатов (Приложение 2. Таблица 1).



6.2 Верификация создания кода блоков

В новой версии для каждой функции языка программирования или блока должна быть проверена и зафиксирована следующая функциональность:

- Подготовка исходных данных для генерации кода в ПО SimInTech согласно требованиям технического задания и рекомендаций типового процесса разработки прикладного ПО (п 5.1.2 Технического задания на модуль генерации кода). Включая следующие требования:
 - Корректность задания исходных данных в среде SimInTech. Новые блоки и функции при использовании в проекте системы управления не должны требовать доработки шаблона генерации кода, кроме существующего (п.п. 5.1.2.1 и 5.1.2.3 Техническое задание на модуль генерации кода SimInTech).
 - Использование базы данных сигналов SimInTech (п.п. 5.1.1.1.2 и 5.1.2.2 Технического задания на модуль генерации кода SimInTech).
 - Использование проекта прикладного SimInTech, для подготовки данных (п.п. 5.1.2.3 Технического задания на модуль генерации кода SimInTech).
 - Возможность математического моделирования проекта SimInTech с новыми блоками и функциями языка программирования (п.п. 5.1.2.3.4 Технического задания SimInTech).
- Генерация исходного кода прикладных программ, согласно требованиям к организации выходных данных (п. 5.1.3 Технического задания на модуль генерации кода).
- Эквивалентность работы прикладного ПО, созданного по исходным кодам и математической модели в ПО SimInTech (п. 5.1.1.3.6 Технического задания на модуль генерации кода).

Последовательность проверки модуля генерации кода:

- Обработка исходных данных.
- Генерация кода.
- Формальная инспекция сгенерированного кода.



6.2.1 Модульное тестирование блоков

В данном тестировании проверяется:

- Работоспособность модуля генерации кода с отдельными блоками или функциями языка программирования.
- Соответствие между блоками и функциями SimInTech и кодом, созданным на языке программирования модулем генерации кода.

Кроме проверки модуля генерации, осуществляется создание исполняемой прикладной программы в виде динамической библиотеки формата Windows для проверки её работы на следующем этапе.

6.2.1.1 Требования к тестовому проекту

Для модульного тестирования создается проект SimInTech согласно требований прикладного ПО (4.2 Типовой процесс разработки ПО) и требований технического задания (п. 5.1.2 Технического задания на модуль генерации кода SimInTech)

Тестовый проект содержит:

- Расчетную схему в составе:
 - Субмодель - группа алгоритмов. Название группы содержит название тестируемого блока или функции. Субмодель (см. рисунок 3), в свою очередь содержит:
 - Оформленный согласно рекомендациям типового процесса разработки прикладного ПО лист алгоритмов. Содержащий таблицу входных сигналов и таблицу выходных сигналов, соединённых с тестируемым блоком или функцией языка программирования (см. рис. 4).
- Базу сигналов проекта. В базе данных должны содержаться:
 - Входные сигналы блока.
 - Выходные сигналы блока.

Структура тестового пакета представлена на рисунке 3.

Пример листа алгоритмов тестового проекта приведен на рисунке 4.

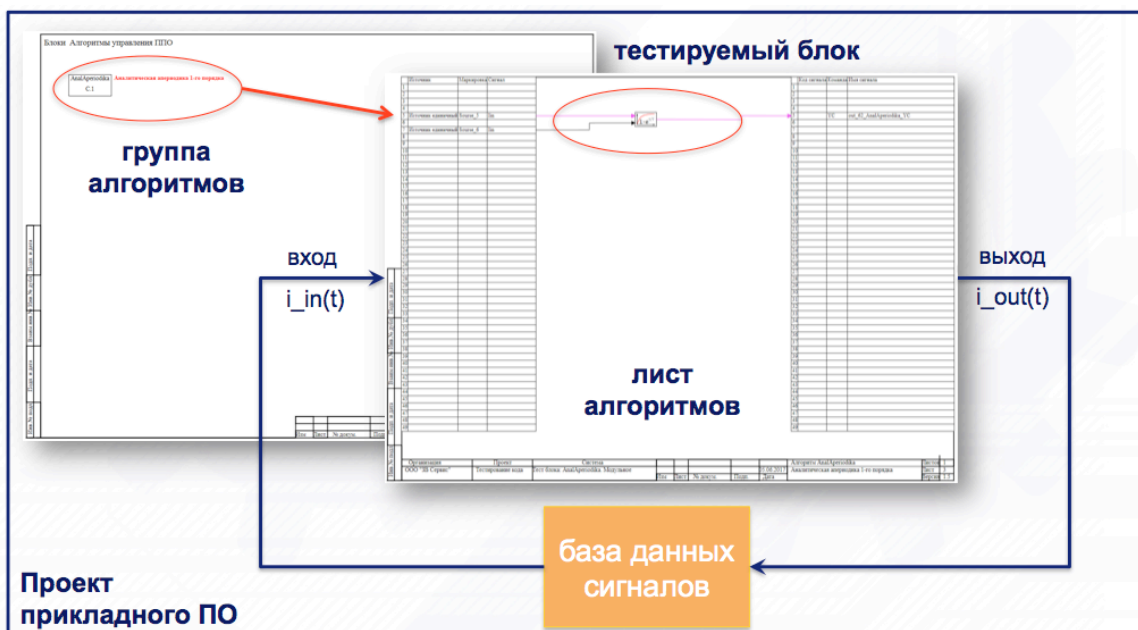


Рисунок 3. Структура пакета тестирования нового блока.

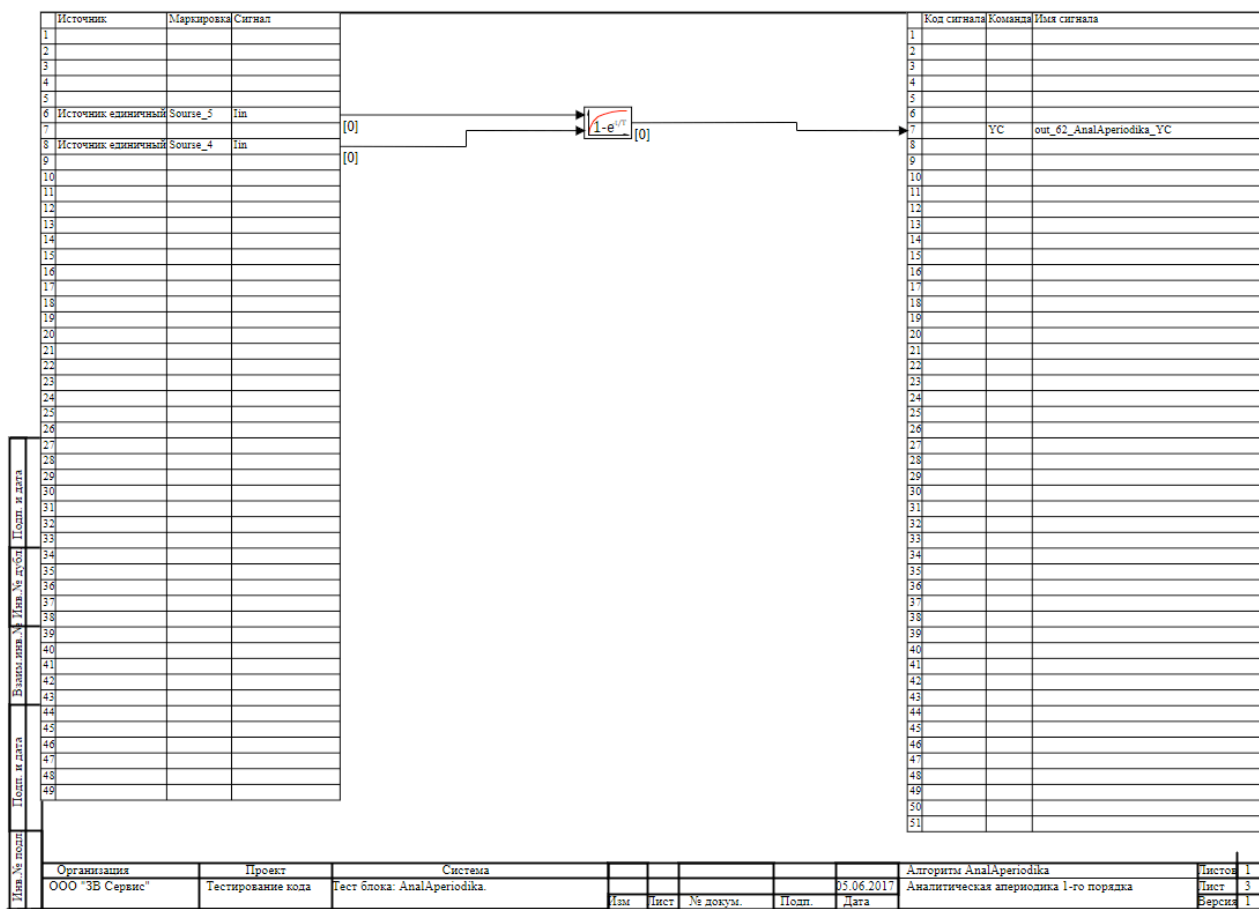


Рисунок 4. Пример структуры листа алгоритма для тестирования блока.



Для формирования входного тестового сигнала в базе данных сигналов используется специальная категория: «Тестовые источники». В категории по умолчанию создаются следующие сигналы 3 типов:

- **Iin** – вещественное число типа double.
- **IinV** – массив вещественных чисел, применяется для проверки блоков, принимающих на вход векторный сигнал.
- **IinM** – матрица – двумерный массив чисел, применяется для проверки блоков, получающих на вход матрицу из действительных чисел.

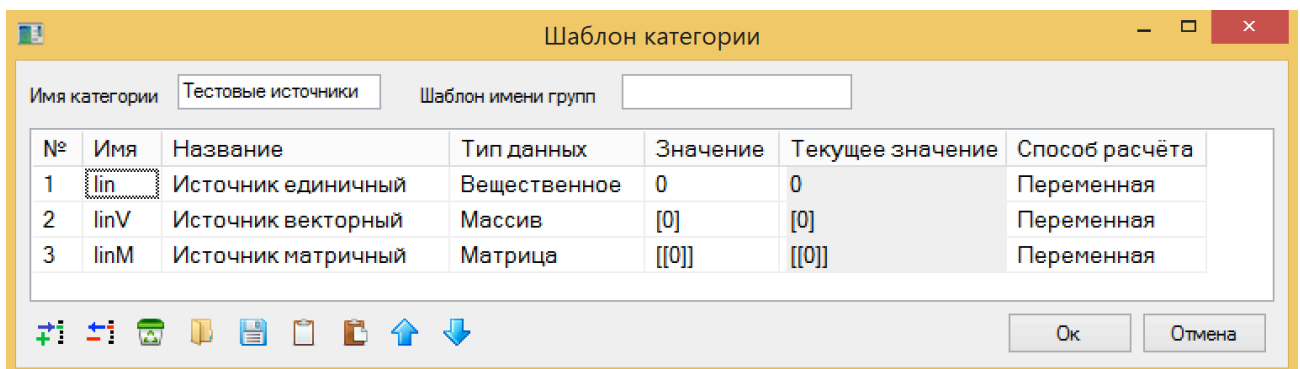


Рисунок 5. Категория базы данных сигналов «Источники»

Таким образом, имя тестового сигнала формируется из имени группы и имени сигнала: например, для единичного сигнала группы **Source_5** имя сигнала будет **Source_5_Iin**.

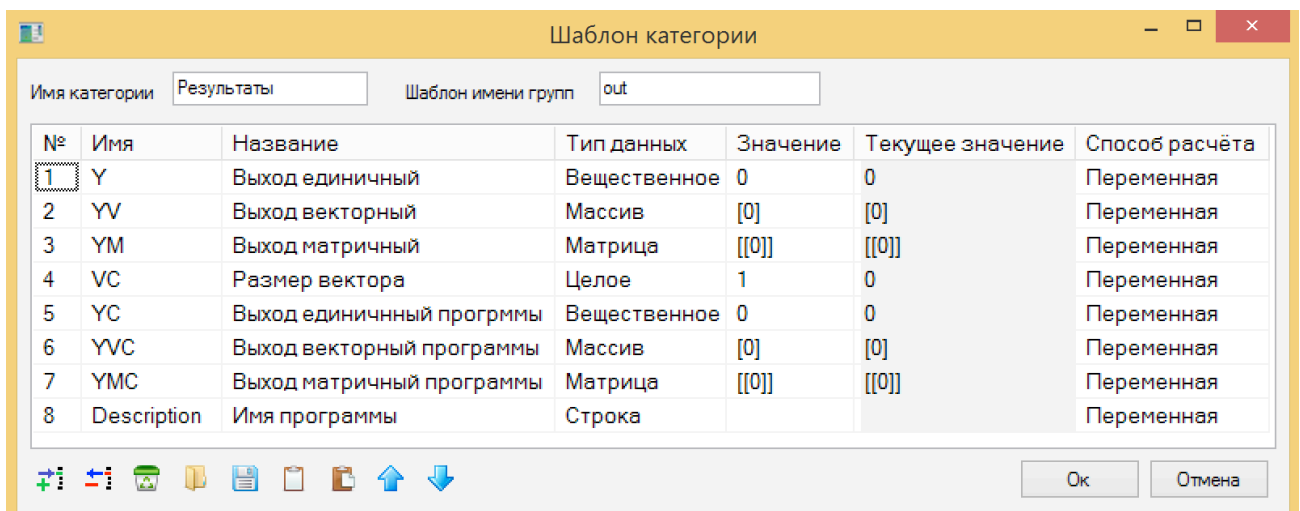


Рисунок 6. Категория «Результаты» для проверки блоков



Для сохранения результатов работы блока в базе сигналов создается структура, обеспечивающая в зависимости от типа блока сохранение выходного сигнала, в переменных разного типа (см. рисунок 6).

При этом в самой структуре предусмотрено хранение как сигнала из блока в тестируемой программе, так и эталонного сигнала, для сравнения на следующем этапе верификации. Для хранения выходного сигнала из тестируемого блока используются сигналы, оканчивающиеся на С:

- Y – вещественный эталонный выход – число типа double.
- YV – массив – эталонный массив чисел, применяется для проверки блоков, выдающих на выход вектор действительных чисел.
- YM – матрица – эталонный двухмерный массив чисел, применяется для проверки блоков, выдающих на выход матрицу из действительных чисел.
- VC – целое число, для хранения размера вектора;
- YC – вещественное число типа double;
- YVC – массив вещественных чисел;
- YMC – матрица – двухмерный массив чисел, применяется для проверки блоков, выдающих на выход матрицу из действительных чисел.



6.2.1.2 Последовательность тестирования

Последовательность тестирования:

1. Открыть тестовый проект для нового блока SimInTech.
2. Убедиться, что проект соответствует требованиям к исходным данным по генерации кода (п. 5.1.3 Технического задания на модуль генерации кода).
3. Запустить проект на расчет. Убедиться, что расчет идет и нет ошибок.
4. Установить в настройках проекта директорию исходников.
5. Установить для параметра **«Директория шаблона кода»**: значение **%codetemplates%MinGW_DLL**
6. В главном меню выбрать пункт: **«Инструменты/Сгенерировать программу»**
7. Убедиться, что в окне сообщений появились сообщения, о завершении создания исходных кодов:

[Информация]: "Исходный текст сохранён в {директория исходных кодов\имя алгоритма}.inc"

[Информация]: "Секция состояний сохранена в {директория исходных кодов\имя алгоритма}_state.inc"

[Информация]: "Заголовок сохранён в {директория исходных кодов\имя алгоритма}.h"

[Информация]: "Заголовок сохранён в {директория исходных кодов\имя алгоритма}_init.inc"

[Информация]: "Исходные тексты программы сгенерированы"

[Информация]: "Используется шаблон кода {директория установки SimInTech}\bin\CodeTemplates\MinGW_DLL\"

[Информация]: "Запуск сборочного скрипта compile.bat "{директория исходных кодов\имя алгоритма}" 192.168.1.1 root root"

8. Убедиться, что в указанной в настройке директории появились файлы, указанные в окнах сообщения.
9. Убедиться, что в директории появился файл с именем **[имя алгоритма].dll**
10. Произвести анализ исходных кодов на соответствие функционалу блока или функции SimInTech, описанном в руководстве пользователя.

**Таблица 2. Результаты второго этапа верификации.**

	Полученные результаты	Выводы	Дальнейшие действия
1	В процессе открытия тестового проекта произошла ошибка и файлы исходных кодов не созданы или созданы не в полном объеме.	Верификация не пройдена.	Отчет об ошибках передается разработчикам.
2	Файлы исходных кодов и dll созданы в полном объеме.	Требования к вводу данных SimInTech выполнены.	Переход к следующему шагу
3	Формальная инспекция исходных кодов, показывает что функционал кода не соответствует описанию в руководстве пользователя SimInTech.	Требуется дополнительная проверка отклонений.	Отклонения фиксируются и передаются разработчику.
4	Формальная инспекция исходных кодов показывает, что функционал созданного кода соответствует описанию в руководстве пользователя SimInTech.	Верификация пройдена.	Переход к следующему этапу верификации.

6.2.2 Артефакты этапа верификации

По результатам данного этапа создается папка, содержащая файлы с исходными кодами, созданными верифицированной версией генератора кода.

А также выпускается отчет, содержащий:

- 1) Схемы тестовых моделей.
- 2) Исходные коды, полученные в результате работы генератора кода.
- 3) Таблица результатов (Приложение 2).



6.3 Тестирование блоков и функций в прикладной программе

Для проверки исходных кодов, созданных модулем генерации кода SimInTech, производится динамический анализ работы программы, созданной в виде динамически подключаемой библиотеки, созданной из тестируемых исходных кодов. Используется метод анимации с программным заданием сигнальных траекторий.

Компиляция из исходного кода исполняемой библиотеки осуществляется на этапе модульного тестирования (см. п. 6.2.1.2).

Созданная библиотека запускается в проекте-загрузчике и тестируется с использованием тестового пакета, как рекомендовано в типовом проекте разработки прикладного ПО.

При необходимости возможно в одном проекте тестирование нескольких программ, подключая несколько dll и создавая несколько эталонных диаграмм алгоритмов.

6.3.1 Требования к тестовому окружению

Тестирование осуществляется с помощью тестового пакета, состоящего из:

- **Тестовый проект SimInTech** – проект, содержащий источники тестовых сигналов и средства для анализа результатов.
- **Проект-загрузчик dll** – проект, содержащий исходную модель для генерации кода, блок-загрузчик dll, обеспечивающий загрузку тестируемой программы в виде dll.
- **Базу данных сигналов** – базу, содержащую все переменные, необходимые для задания сигнальных траекторий и чтения результатов работы программы.

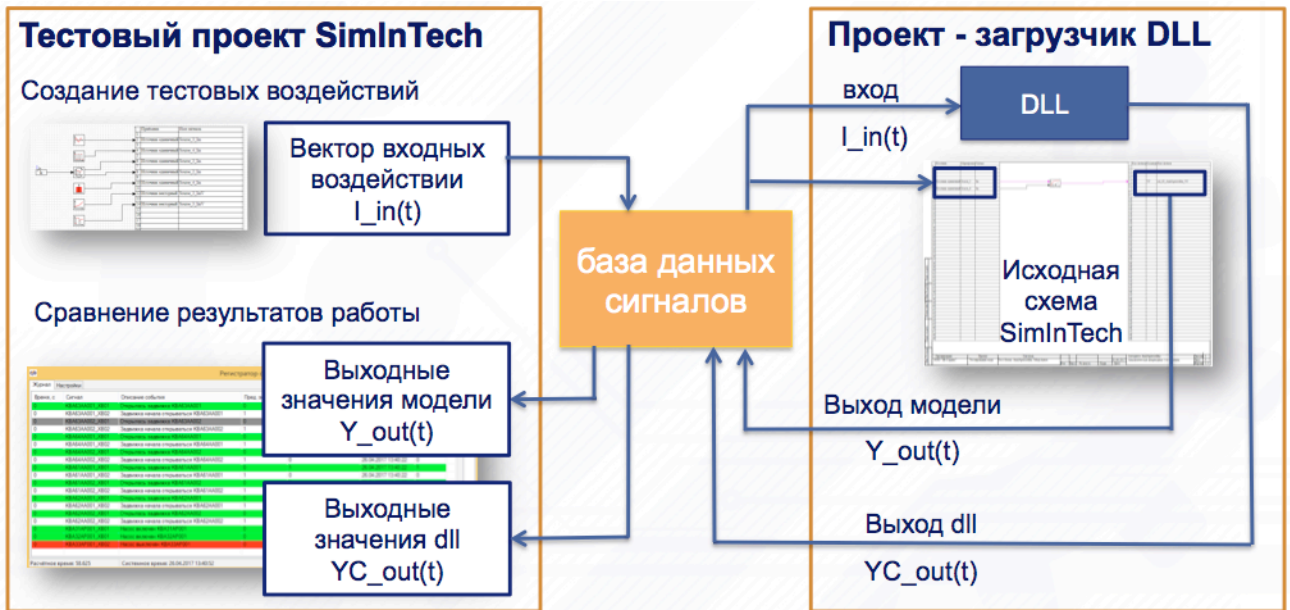


Рисунок 7. Структура тестового окружения



6.3.1.1 Проект-загрузчик dll

Проект-загрузчик DLL содержит:

- Специальный блок «**Внешняя dll**», используется для запуска тестируемой dll. В данном блоке указывается имя dll, созданной на этапе модульного тестирования.
- Субмодель, со схемой SimInTech (эталонная модель), полностью аналогичная той, из которой происходит генерация кода для dll.
- База данных сигналов для чтения входных воздействий, является той же структурой, которая использовалась при генерации кода (см. рисунок 5), для записи используется та же структура, что и для генерации кода (см. рисунок 6).

6.3.1.2 Тестовый проект

Тестовый проект содержит:

- Генератор входных воздействий для тестируемых блоков и функций. Входные воздействия подбираются таким образом, чтобы обеспечить тестирование работы блока в максимально возможных режимах работы.
- Блок сравнения. Алгоритм верификации, обеспечивает сравнение двух сигналов, анализ расхождения и формирование отчета по результатам верификации.
- База данных сигналов (единая с проектом-загрузчиком) с входными воздействиями и выходными значениями. Категория базы данных для записи тестового воздействия аналогична структуре, использованной для модульного тестирования (см. Рисунок 6). Для чтения результатов используется структура, аналогичная структуре, созданной для модульного тестирования (см. Рисунок 7).
- Регистратор событий, обеспечивающий регистрацию отклонения сигналов больше допустимой.



6.3.1.2.1 Блок сравнения

Представляет собой субмодель SimInTech, которая осуществляет подключение к базе данных сигналов, чтение результатов работы dll и эталонной схемы, сравнение результатов и формирование отчета. Схема алгоритма сравнения представлена на рисунке 9.

В качестве параметров данного блока используются:

- resultname – имя структуры в базе данных сигналов тестового пакета.
- tolerance – допустимая абсолютная погрешность по модулю.

Название	Имя	Значение	Текущее значение
Результаты	resultname	out_168_Rele2_ReleContact2	out_168_Rele2_ReleContact2
Допустимая погрешность	tolerance	1e-12	1E-12

Рисунок 8. Параметры блока сравнения единичного сигнала

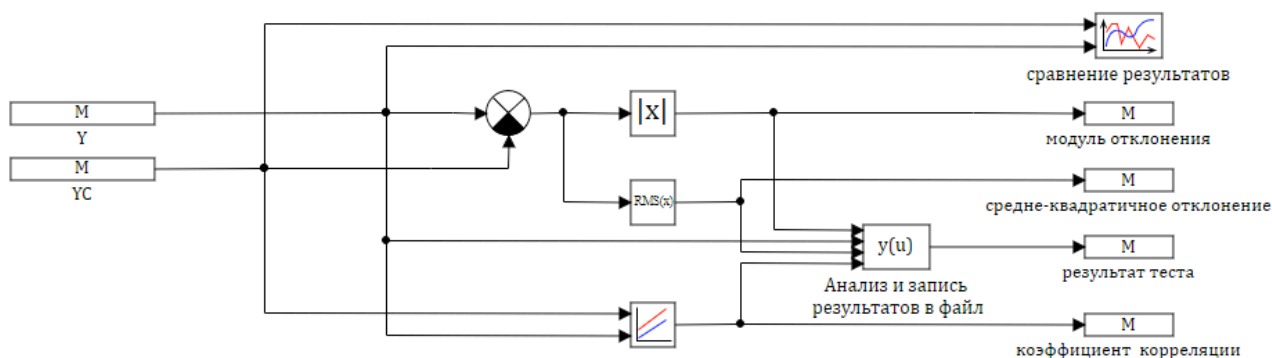


Рисунок 9. Схема алгоритма сравнения единичных сигналов

На вход в схему берутся два сигнала из заданной структуры базы данных:

- Y – выход модели SimInTech.
- YC – выход тестируемой dll. При необходимости для блоков, у которых запись из базы данных в dll происходит на шаг раньше или позже возможно указать в настройках шаг чтения сигналов.

Для визуального сравнения используется график «Сравнение результатов».

С помощью специальных блоков SimInTech в процессе тестирования выполняется вычисление следующих величин:

- Абсолютное отклонение по модулю.
- Средне-квадратичное отклонение величины.



- Коэффициент корреляции сигналов.

Для анализа результатов тестирования используется блок, который обеспечивает сравнение отклонения с заданным. В скрипте осуществляется вычисление абсолютного максимального отклонения и формируется файл отчета о результатах тестирования. Сообщения об отклонениях выводятся также в окно сообщений в нижней части программы.

Блок представляет собой скрипт на встроенном языке программирования SimInTech. Текст скрипта приведен в Приложении 1. На каждом шаге расчета осуществляется оценка совпадения результатов эталонной модели и программы и выводится в специальную переменную TestPassed.

Отчет сохраняется в текстовый файл с именем **[resultname].dat**, где **resultname** – имя структуры с результатами моделирования в базе данных сигналов.

Пример файла отчета об успешном завершении тестирования:

Анализ результата тестирования для структуры out_168_Rele2_ReleContact2

Модельное время расчета: 173.1

Минимальное значение: 0

Максимальное значение: 1

Модуль максимального отклонения: 0

Среднеквадратичное отклонение: 0

Коэффициент корреляции: 1

Тестирование пройдено заданная точность обеспечивается.

Пример текста файла отчета о неудачном завершении тестирования:

Анализ результата тестирования для структуры out_4_Sin

Модельное время расчета: 33.86

Минимальное значение: -1

Максимальное значение: 1

Модуль максимального отклонения: 0.001

Среднеквадратичное отклонение: 0.0001549

Коэффициент корреляции: 1

Отклонение больше допустимого !



6.3.1.2.2 Структура базы данных сигналов для тестирования

Для сохранения результатов тестирования в базе данных сигналов создается расширенная структура данных на основе структуры, используемой для хранения результатов работы тестируемой программы и эталонной модели.

К существующим сигналам добавляются сигналы, описывающие результаты анализа сравнения выхода программы и эталонного блока (см. рисунок 10):

- Maxdev – максимальное отклонение;
- RMS – среднеквадратичное отклонение;
- Ks – коэффициент корреляции;
- TestPassed – отметка о выполнении.

Шаблон категории










Имя категории

Результаты

Шаблон имени групп

out

№	Имя	Название	Тип данных	Значение	Текущее знач	Способ расчёта
1	Y	Выход единичный	Вещественное	0	0	Переменная
2	YV	Выход векторный	Массив	[0]	[0]	Переменная
3	YM	Выход матричный	Матрица	[[0]]	[[0]]	Переменная
4	VC	Размер вектора	Целое	1	0	Переменная
5	YC	Выход единичный программы	Вещественное	0	0	Переменная
6	YVC	Выход векторный программы	Массив	[0]	[0]	Переменная
7	YMC	Выход матричный программы	Матрица	[[0]]	[[0]]	Переменная
8	Description	Описание	Строка	Тест блока	Тест блока	Переменная
9	maxdev	Максимальное отклонение	Вещественное	0	0	Переменная
10	RMS	Среднеквадратичное отклонение	Вещественное	0	0	Переменная
11	ks	Коэффициент корреляции	Вещественное	0	0	Переменная
12	TestPassed	Отметка о выполнение теста	Двоичное	Нет	Нет	Переменная



OkОтмена

Рисунок 10. Структура базы данных сигналов для сохранения результатов теста

6.3.1.2.3 Регистратор событий.

Регистратор событий обеспечивает отслеживание изменения параметра TestPassed. В случае выхода за пределы точности осуществляется фиксация события.

Возврат значения в заданные пределы также фиксируется.

При моделировании работы программы в различных проектах одного пакета возможна ситуация, когда запись сигнала в базу данных осуществляется раньше, чем завершается шаг интегрирования задачи. В этом случае возможна ситуация, когда



отклонение сигнала в блоке сравнения будет связано со сдвигом шага интегрирования тестового пакета при корректной работе программы в виде dll. В этом случае в регистраторе событий появится два события, одно об выходе отклонения за допустимые пределы, другое о возврате в заданные пределы.

Такие отклонения фиксируются регистратором событий в виде двух записей с одним и тем же временем и при анализе отклонений не должны учитываться как ошибки. См. рисунок 11.

Время, с	Сигнал	Описание события	Пред. значение	Новое значение
1169.19799997926	out_168_Rele2_ReleContact2_TestPassed	Отклонение выше допустим	1	0
1169.19899997926	out_168_Rele2_ReleContact2_TestPassed	Отклонение в пределах норм	0	1
1956.68999996064	out_168_Rele2_ReleContact2_TestPassed	Отклонение выше допустим	1	0
1956.69099996064	out_168_Rele2_ReleContact2_TestPassed	Отклонение в пределах норм	0	1
3915.00200033884	out_168_Rele2_ReleContact2_TestPassed	Отклонение выше допустим	1	0
3915.00300033884	out_168_Rele2_ReleContact2_TestPassed	Отклонение в пределах норм	0	1
5058.4890005718	out_168_Rele2_ReleContact2_TestPassed	Отклонение выше допустим	1	0
5058.4900005718	out_168_Rele2_ReleContact2_TestPassed	Отклонение в пределах норм	0	1

Расчётное время: 7049.071 Системное время: 22.08.2017 19:43:31

Рисунок 11. Регистратор событий для тестового проекта.



6.3.2 Последовательность тестирования

Входные воздействия создаются с помощью тестового проекта SimInTech и записываются в базу данных сигналов на каждом шаге синхронизации.

В проекте-загрузчике входные воздействия передаются в dll и в эталонную модель.

Средствами математического ядра выполняется динамический расчет эталонной модели. Программа в dll, созданная на основе сгенерированных кодов, выполняет расчет выходных значений.

По завершению шага расчета результаты записываются в базу данных сигналов.

В тестовом проекте осуществляется чтение результатов работы dll и модели, их сравнение и анализ.

По завершению тестового моделирования формируются текстовые файлы отчета.

Записи регистратора событий сохраняются в текстовый файл.

Таблица 3. Результаты третьего этапа верификации.

	Полученные результаты	Выводы	Дальнейшие действия
1	В процессе открытия тестового пакета произошла ошибка.	Верификация не пройдена.	Отчет об ошибках передается разработчикам.
2	Тестовый пакет открылся и все пакеты загрузились	Данные для моделирования подготовлены верно	Переход к динамическому моделированию
3	В процессе запуска на расчет произошла ошибка.	Верификация не пройдена.	Данные об ошибке передаются разработчику.
4	В процессе проект запустился на расчет в процессе моделирования появились сообщения об ошибках.	Верификация не пройдена.	Данные об ошибке передаются разработчику
5	Расчет завершился по достижению	Верификация не	Данные об ошибках



	заданного времени моделирования. Есть сообщения о превышении заданной точности в течении длительного периода времени.	пройдена.	передаются разработчику
6	Расчет завершился по достижению заданного времени моделирования. Сообщения об ошибках нет.	Верификация пройдена	Отклонения фиксируются и передаются разработчику.

6.3.3 Артефакты этапа верификации

Набор текстовых файлов с отчетами по результатам сравнения работы модели в SimInTech и программы, созданной на основе исходных кодов.

Файл регистратора событий тестового проекта.

Отчет, содержащий:

- Описание тестового рабочего места.
- Результаты верификации (таблица 3 Приложения 2).



ПРИЛОЖЕНИЕ 1. Текст блока анализа результатов тестирования

```
input u, YC, RMSdev, kc;
output y;
var
filename:string = resultname + ".dat",
s1:string = "ng = resultname + ".dat",ванияring = "ng = resultresultname +chr(13)+chr(10),
s2:string = "string3",
Maxdev = 0,
MaxY = 0,
MinY = 0;

initialization
//itIALIZATION
f_id = createfile(filename, -1);
//id = createfile(filename, -1);
writetext(f_id, s1);
Y0=YC;
y = 0;
end;

Maxdev = max(Maxdev, u);
MaxY = max(MaxY, YC);
MinY = min (MinY, YC);

finalization
s2 = "zationMinY, YC);расчетаationMfloattostrf(time, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
s2 = "ext(f_id, s2);ачениеext(f_floattostrf(MinY, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
s2 = "ext(f_id, s2);начениеext(f_ifloattostrf(MaxY, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
s2 = "ext(f_id, s2);льногоext(f_id, s2); + floattostrf(Maxdev, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
s2 = "ext(f_id, s2);чное "ext(f_id, s2); floattostrf(RMSdev, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
s2 = "Коэффициент корреляции: " + floattostrf(kc, 0, 4, 0)+chr(13)+chr(10);
writetext(f_id, s2);
```



```
if (Maxdev <= tolerance) then
begin
y = 1;
s2 = "Тестирование успешно заданная точность обеспечивается" +chr(13)+chr(10);
end else
begin
s2 = "Отклонение больше допустимого !" +chr(13)+chr(10);
seterrorstr("отклонение больше допустимого",1,submodel);
end;
writetext(f_id, s2);
freeobject(f_id);
end;
```



ПРИЛОЖЕНИЕ 2. Примеры таблиц результатов

Таблица 1 Результат тестирования для существующих блоков.

Наименование блока/функции	Эквивалентность исходного кода с предыдущей версией	Примечание	Результат верификации

Таблица 2 Формальная инспекция новых блоков.

Наименование блока/функции	Формальная инспекция кода	Примечание	Результат верификации

Таблица3. Сравнение модели и программы.

Наименование блока/функции	Сравнение работы	Примечание	Результат верификации