|  |  |
| --- | --- |
|  | ООО «3В Сервис»  РФ, 127051, Москва, ул. Трубная 25 стр 1 офис 6  Тел./ф (495) 221-22-53  [www.3v-services.com](http://www.3v-services.com/) |

**Утверждаю**

генеральный директор

ООО «ЗВ Сервис»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Петухов В.Н.

****

**Среда динамического моделирования технических систем SimInTech™**

**Типовой процесс создания прикладного программного обеспечения для систем контроля и управления важных для безопасности АЭС**

Модуль генерации кода для систем реального времени

ШИФР ГК16ТП

Москва, 2016

# **Аннотация**

В данном документе описывается рекомендуемый типовой процесс создания прикладного программного обеспечения для систем контроля и управления важных для безопасности АЭС с использованием программного обеспечения «Среда динамического моделирования технических систем SimInTech».

Пользователи SimInTech могут использовать этот процесс как часть модельно-ориентированного жизненного цикла проектирования прикладного ПО с использованием генерации кода.

Процесс модельно-ориентированного проектирования прикладного ПО в SimInTech обеспечивает автоматическую генерацию кода промышленного качества из исполняемых графических моделей, созданных на предметно-ориентированном языке программирования SimInTech. Данный код может быть использован для создания прикладного ПО в системах контроля и управления важных для безопасности АЭС.

SimInTech обеспечивает графическое моделирование в нотации функционально- блочных диаграмм и диаграмм состояний конечных автоматов. Сгенерированный код прикладного программного обеспечения является верифицированным на соответствие между расчетной схемой и исходной моделью.

Использование математической модели объекта управления обеспечивает верификацию программного обеспечения на предмет соответствия функциональности проектируемого ПО требованиям со стороны конструктора и технолога-проектировщика.

Средствами математического моделирования SimInTech можно проверить соответствие между исходной расчетной моделью в SimInTech и исполняемым кодом прикладного программного обеспечения, путем задания идентичных входных воздействий и анализа результата работы. Это обеспечивает дополнительную проверку технологической цепочки по преобразованию исходного кода в исполняемое ПО, включающее процессы и инструменты компиляции исходных кодов.

В данном документе даются рекомендации по настройке SimInTech и применению специализированных инструментов среды, для реализации жизненного цикла разработки программного согласно требованиям ГОСТ Р МЭК 60880 – 2011.

СОДЕРЖАНИЕ

**Аннотация** 2

1. **Введение** 5

**2.** **Рекомендации стандарта ГОСТ Р МЭК 60880 – 2011** 6

**3.** **Термины и определения** 8

**4.** **Типовой процесс проектирования в SimInTech** 15

**4.1** **Этапы разработки в ПО** 16

**4.2 Рекомендуемая структура прикладного ПО** 18

**5.** **Проектирование базы данных сигналов** 19

5.1 Исходные данные 19

5.2 Спецификация требований к прикладному ПО 20

5.3 Принципы проектирования 21

5.4 Структура базы данных 22

5.4.1 Рекомендации по созданию структуры базы данных 23

5.5 Наименование переменных и система кодирования 26

5.6 Разграничение доступа к базе данных сигналов 27

**6.** **Проектирование расчетной схемы в SimInTech** 28

6.1 Содержание расчетной схемы 30

6.1.1 Оформление проекта прикладного ПО 30

6.1.2 Оформление листов алгоритмов 30

6.1.3 Разграничение прав доступа к проекту. 31

6.2 Структура прикладного ПО на вычислительном приборе 32

6.2.1 Входные данные прикладного ПО 33

6.2.2 Обработка данных 34

6.2.2.1 Самодиагностика ввода данных 34

6.2.3 Функциональное ПО 36

6.2.4 Вывод данных 37

**7.** **Функциональная верификация и валидация** 38

7.1 Статический анализ проекта в SimInTech 39

7.1.1 Статический анализ базы данных сигналов. 39

7.1.2 Статический анализ расчетной схемы проекта 40

7.2 Динамический анализ расчетной схемы 41

7.2.1 Структура тестового пакета 42

7.2.2 Варианты создания вектора тестового воздействия 45

7.2.2.1 Тестовые циклограммы 45

7.2.2.2 Ручное изменение входных параметров 45

7.2.2.3 Данные моделирования из файла 46

7.2.2.4 Математическое моделирование систем АЭС 46

7.2.3 Анализ работы прикладного ПО в тестовом проекте 48

7.2.3.1 Тестовое покрытие прикладного кода 49

7.2.3.1.1 Тестовое покрытие входных переменных 49

7.2.3.1.2 Тестовое покрытие выходных переменных 49

7.2.3.1.3 Тестовое покрытие ветвления и логических переключений 49

**8.** **Генерация кода прикладного ПО** 51

8.1 Настройка генерации кода в SimInTech 51

8.2 Верификация технических средств системы контроля и управления 52

8.2.1 Полигон тестирования СКУ 52

**9.** **Соответствие требованиям ГОСТ Р МЭК 60880-2011 и типового процесса разработки прикладного ПО в среде SimInTech** 54

# **Введение**

Среда динамического моделирования технических систем SimInTech может быть использована для проектирования и создания программного обеспечения систем управления, в качестве инструментального средства для разработки прикладного программного обеспечения.

Графические средства позволяют обеспечить создание спецификации требований к программному обеспечению, а математическое ядро обеспечивает моделирование и динамический анализ данных требований, для осуществления процессов верификации и валидации системы.

Модуль генерации кода обеспечивает автоматическое создание исходного кода прикладного программного обеспечения, подготовленного к компиляции средствами выбранной программно-аппаратной платформы целевой системы.

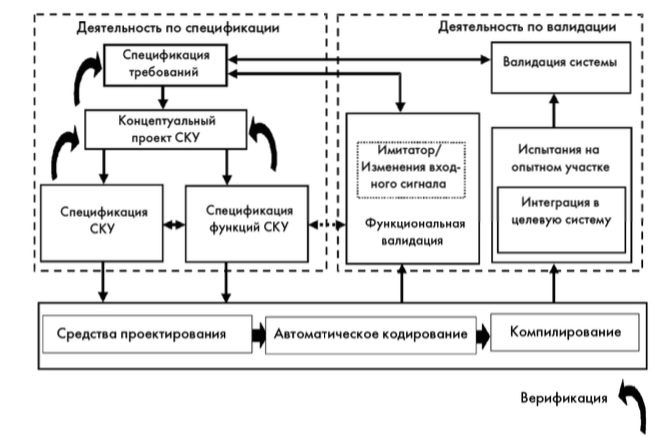
# **Рекомендации стандарта ГОСТ Р МЭК 60880 – 2011**

SimInTech обеспечивает создание прикладного программного обеспечения в виде набора графических функционально-блочных диаграмм, что соответствует применению предметно- ориентированного языка программирования согласно определения ГОСТ Р МЭК 60880 – 2011.

«Среда динамического моделирования технических систем SimInTech» вместе с модулем генерации кода обеспечивает поддержку (полностью или частично) следующих мероприятий из жизненного цикла безопасности системы [Раздел 5.1 Общая информация]:

1. Проектирование прикладного ПО.
2. Разработка/Генерация прикладного ПО.
3. Верификация ПО.

В общем виде жизненный цикл для технологии прикладного программирования по рекомендациям ГОСТ Р МЭК 60880 – 2011 представлен на рисунке 1.



**Рисунок 1. Жизненный цикл для технологии прикладного программирования.**

# **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** **Видеокадр (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 связных 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.

П р и м е ч а н и е – 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.

П р и м е ч а н и е – 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.

П р и м е ч а н и е – 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.

П р и м е ч а н и е 1 – Программное обеспечение операционной системы групп оборудования обычно реализуется с использованием универсальных языков. П р и м е ч а н и е 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.

П р и м е ч а н и е – 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.

П р и м е ч а н и е – 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.

# **Типовой процесс проектирования в SimInTech**

Для разработки прикладного программного обеспечения систем контроля и управления важных для безопасности АЭС, SimInTech используется как средство проектирования и автоматического кодирования (см. рис. 1). Предметно-ориентированный язык SimInTech обеспечивает создание программного обеспечения в виде функционально-блочной диаграммы (схемы). Это позволяет описывать функциональность проектируемого ПО в наглядном виде, понятном для широкого круга специалистов в области проектирования АЭС.

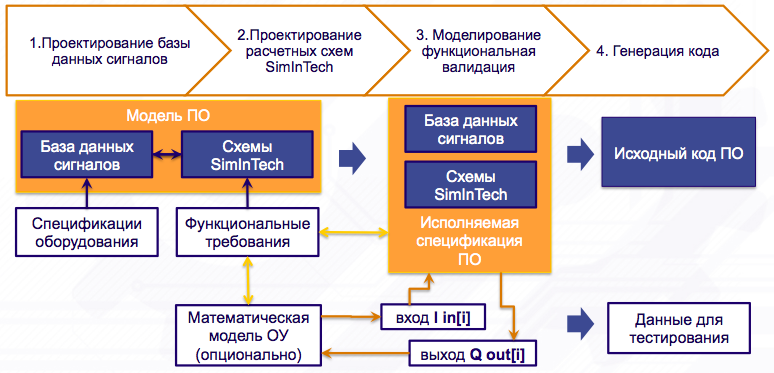
Схема SimInTech обеспечивает первичное представление функциональности ПО. В процессе проектирования схема уточняется, дополняется функциональностью и проектными данными и в результате последовательного процесса создается полная спецификация ПО, которая является исполняемой (моделируемой) в среде SimInTech и используется для генерации кода.

В процессе разработки рекомендуется осуществлять динамическое моделирование схемы SimInTech на каждом этапе уточнения или изменения схемы для верификации создаваемого ПО, на предмет соответствия требованиям.

Для более надежной верификации прикладного ПО, рекомендуется создавать в SimInTech модель векторов входных воздействий для прикладного ПО (тестовые последовательности), для использования их на этапе испытаний аппаратуры СКУ.

## **Этапы разработки ПО**

Для соответствия требованиям стандарта ГОСТ Р МЭК 60880 2011 рекомендуется использовать типовой процесс создания прикладного ПО, изображенный на рисунке 2. Данный процесс является уточненной частью общего процесса (рис. 1).

****

**Рисунок 2. Типовой процесс создания прикладного ПО в SimInTech**

Проектирование и реализация ПО должна происходить в 4 этапа:

**Этап 1.** Создание базы данных сигналов – процесс формирования всех необходимых программе данных, на основе спецификаций оборудования и требований к прикладному ПО.

**Этап 2.** Создание набора схем SimInTech – процесс формирования алгоритмов управления, логики работы, функционала ПО в SimInTech.

**Этап 3.** Моделирование и функциональная валидация – процесс моделирования в SimInTech, который является процессом динамического анализа созданного ПО с заданием входных воздействии **I in[i]** и анализом результатов работы программы **Q out[i]**, для проверки соответствия требованиям. Таким образом, SimInTech используется для создания имитатора входных сигналов (см. рис.1)

*Вектор входных воздействий может быть получен путем создания математической модели АЭС или отдельных систем АЭС, для которых разрабатывается СКУ, в среде SimInTech. При использовании модели объекта вектор выходных значений используется в модели для динамического расчета поведения объекта и системы управления в различных режимах работы.*

Векторы входных сигналов и выходные данные рекомендуется сохранять и использовать для тестирования ПО, установленного на целевую систему на опытном участке (см. рисунок 1). В варианте создания математической модели объекта управления она так же может быть использована для на опытном участке.

**Этап 4.** Автоматическая генерация кода – создание исходного кода ПО, готового к созданию объектного кода и загрузке на вычислительный компьютер СКУ. Для формирования исходного кода должны использоваться готовые шаблоны, обеспечивающие автоматическую подготовку исходного кода, для обработки инструментальными средствами (компиляция исполняемых и динамических библиотек) и интеграции с системным ПО.

*Создание шаблонов для генерации кода является отдельным этапом и не описывается в данном типовом процессе.*

Этапы 1 – 4 являются универсальными и не зависят от системного ПО и аппаратной составляющей СКУ. Компилирование объектного кода и интеграция прикладного ПО с системным ПО на вычислительные приборы выполняется инструментальными средствами, выбранными для конкретной программно-аппаратной платформы.

При разделении ПО на структуры см. п. 4.2 этапы 1-3 рекомендуется выполнять для каждой части ПО.

Все этапы проектирования программного обеспечения (ПО) в среде SimInTech должны выполняться согласно рекомендациям Приложения В. ГОСТ Р МЭК 60880 – 2011

## **4.2 Рекомендуемая структура прикладного ПО**

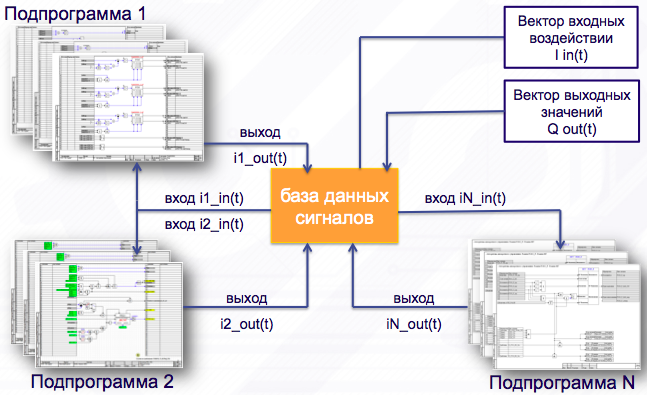
Прикладное ПО в среде SimInTech формируется в виде проектов. Необходимо создавать отдельные проекты SimInTech для:

* прикладного ПО, исполняемого на разных вычислительных узлах в распределённой системе управления;
* частей функционального ПО, исполняемых на одном компьютере с разными тактами выполнения.

Проект программного обеспечения в среде SimInTech должен создаваться в виде набора подпрограмм (субстуркутр), оформленных в виде листов алгоритмов. Листы алгоритмов объединяются в структуры «Группы алгоритмов».

Каждая из субструктур в среде SimInTech может разрабатываться и модифицироваться независимо от остальных подпрограмм.

Обмен данными между субструктурами (подпрограммами) осуществляется с помощью базы данных сигналов. Общая рекомендуемая схема проекта прикладного программного обеспечения представлена на рисунке 3.



**Рисунок 3. Общая схема проекта ПО в SimInTech**

Для целей верификации вектор входных воздействий может быть получен путем динамического моделирования работы технологической системы средствами SimInTech.

Вектор выходных значений может получен путем моделирования работы ПО математическим ядром SimInTech.

Для каждой из подпрограмм должны выполнятся этапы разработки 1 – 3, с тестированием каждой из разработанной подпрограммы и финальным комплексным тестированием интегрированного прикладного ПО.

После комплексного тестирования производится генерация исходного кода на универсальном языке программирования для создания объектного файла и компиляции исполняемой программы для аппаратных средств.

# **Проектирование базы данных сигналов**

## Исходные данные

Исходными данными для проектирования базы данных сигналов являются спецификация оборудования СКУ, спецификация функций СКУ (см. рис.1). Спецификация формируется на основании проектных данных АЭС. P&ID диаграммы технологических схем АЭС служат для получения данных о технологическом оборудовании, которое управляется СКУ.

Возможно автоматическое заполнение базы данных сигналов из информационных систем, применяемых для проектирования АЭС.

Основными данными для формирования структуры базы данных сигналов являются:

* Технологическая схема системы АЭС, для которой выполняется проектирование СКУ. Схема должна содержать описание точек контроля и исполнительных органов.
* Спецификация аппаратных средств СКУ, содержащее описание аппаратной реализации СКУ, платы ввода-вывода, вычислители.
* Системное программное обеспечение (программные интерфейсы, протоколы обмена данными).
* Описание алгоритмов управления, регламенты и требования к функциям СКУ.

## Спецификация требований к прикладному ПО

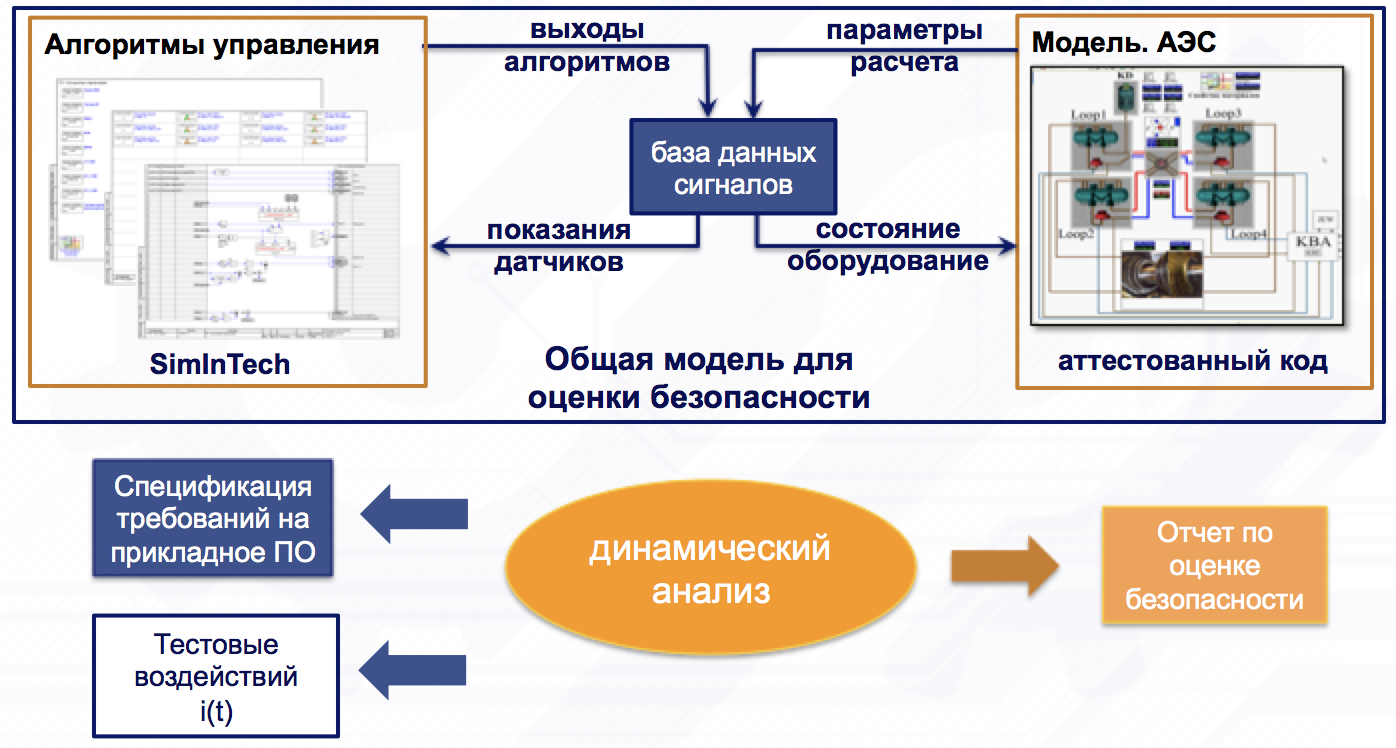
Расчетная схема SimInTech является предметно-ориентированным языком программирования и может быть использована для формирования спецификации требований к прикладному ПО.

Особенностью проектирования систем АЭС, важных для безопасности является необходимость создавать математическую модель систем для анализа безопасности.

На стадии проектирования технологических систем рекомендуется использовать SimInTech для создания комплексной модели объекта. При этом в SimInTech создаются расчетные схемы алгоритмов управления, а для моделирования технологических систем используются аттестованные соответствующие органами расчетные коды.

Модель системы управления в SimInTech, созданная проектировщиком для выполнения анализа безопасности, является источником спецификации требований к прикладному ПО. Динамическая модель, созданная с помощью аттестованных расчетных кодов является источником тестовых воздействий для процедуры верификации проекта алгоритмов.

Рекомендуется разрабатывать динамическую модель алгоритмов таким образом, чтобы алгоритмы, созданные на стадии проектирования могли использоваться в прикладном ПО без изменения или с минимальными доработками.



**Рисунок 4. Спецификация ПО и расчетная модель АЭС**.

## 5.3 Принципы проектирования

База данных сигналов должна быть первоисточником данных по переменным проекта для расчетных схем прикладного ПО.

Проектирование базы данных сигналов должно идти «сверху вниз», от общего к частному. Сначала формируется структура объектов, потом вносятся экземпляры объектов, затем вносятся параметры конкретных объектов.

Структура базы данных должна создаваться на ранней стадии проекта и по возможности не меняться на последующих стадиях проектирования.

База данных сигналов SimInTech должна содержать структурированные списки переменных, сгруппированные по одному или нескольким признакам.

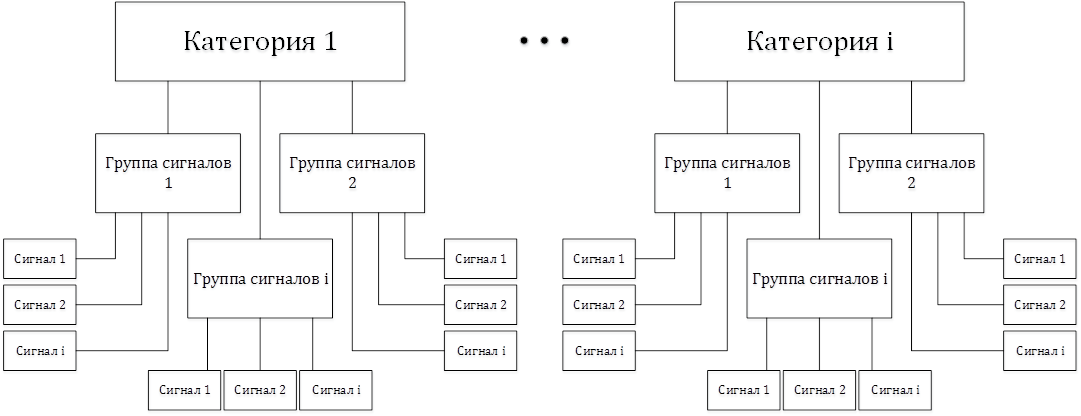
## 5.4 Структура базы данных сигналов

Элементарной единицей базы сигналов является сигнал. Сигнал – переменная одного из доступных типов данных, несущая какую-либо информацию. Сигнал в SimInTech обладает рядом свойств (имя, название, тип данных, значение, способ расчета), набор которых является персональным для каждого сигнала и также хранится в базе.

Для структурирования сигналов и обеспечения удобства поиска и доступа к ним, применяются объединение сигналов в структуру «Группа сигналов» и объединение групп сигналов в структуру «Категории». Такое объединение является аналогией моделируемым аппаратным сигналам, где категорией можно считать тип оборудования, а группой сигналов – конкретную аппаратную часть, конкретный экземпляр данного типа оборудования, внутри которой используется сигнал. Например, сигнал включения насоса с номером 1 будет располагаться в категории «Насосы» и в группе сигналов «Насос1».

Этот принцип организации базы сигналов не является жестко регламентируемым, но именно такая концепция формирования базы закладывалась разработчиками и именно ею рекомендуется руководствоваться при формировании и наполнении базы сигналов.

Таким образом, структура базы сигналов в общем виде выглядит так как показано на рисунке 5:



**Рисунок 5. Структура базы данных сигналов**.

## 5.4.1 Рекомендации по созданию структуры базы данных

Рекомендуемые признаки для структурирования переменных в базе данных сигналов.

* Список точек контроля и исполнительных механизмов из технологической схемы системы, для которой создается проект.
* Спецификация программно-аппаратной части СКУ (платы ввода-вывода, компьютеры, внешние интерфейсы).
* Типовое оборудование, повторяющиеся в схеме (задвижки, клапана, двигатели, регуляторы).
* Типовые (повторяющиеся) подпрограммы управления оборудованием (блок управления клапаном, блок управления задвижкой, ПИД-регуляторы и т.п.).
* Видеокадры системы управления верхнего уровня.
* Каналы управления системой;
* Функциональные части программного обеспечения (защиты, уставки, сигнализация).

База данных сигналов в SimInTech имеет трехуровневую структуру:

1 уровень – **Категории**. Представляет собой список категорий, определенных для конкретного проекта ПО. Каждая категория содержит список шаблонных сигналов (переменных) определенных для данной категории в проекте ПО. В терминах объектно-ориентированного программирования, категория это ***имя класса***.

*Примеры: «БУЗ» (блок управления задвижкой), «БУД» (блок управления двигателем), «Датчики», «БВД-32» (блок ввода данных) и т.п.*

2 уровень – **Группы сигналов.** Представляет собой список именованных групп сигналов для каждой категории 1 уровня. Определяет конкретный существующий в проекте объект, описываемый переменными, заданными в категории. Для наименования групп сигналов необходимо использовать только латинские буквы. В терминах объектно-ориентированного программирования это ***имя экземпляра класса.***

3 уровень – **Сигналы и данные для групп**. Представляет собой список сигналов (переменных) проекта, относящихся к конкретной группе сигналов. В терминах объектно-ориентированного программирования, это ***поле класса***.

Каждая переменная в базе данных сигналов описывается следующими полями:

**Имя** – уникальное в пределах группы имя сигнала. Для наименования сигналов рекомендуется использовать английские буквы и цифры.

**Название** – произвольное описание сигнала на русском языке.

**Тип данных** – один из возможных в SimInTech типов данных:

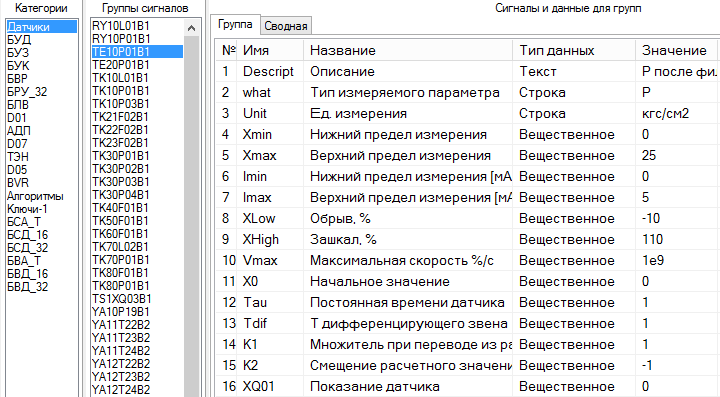
*Вещественное, Целое, Двоичное, Цвет, Строка, Массив, Растровое изображение и др.*

**Значение** – начальное интерпретируемое значение сигнала (на момент создания или загрузки базы данных при инициализации моделирования).

**Текущее значение** – значение сигнала на текущий момент времени.

**Способ расчета** – *константа, переменная, заблокирована.*

На рисунке 5. Приведен пример заполнения базы данных сигналов для группы датчики.



**Рисунок 6. Пример заполнения базы данных для категории «Датчики».**

Таким образом, база данных сигналов обеспечивает наглядное описание всех переменных проекта ПО в виде, доступном для анализа специалистам предметной области без опыта работы с универсальными языками программирования.

База данных сигналов содержит описание всех переменных проекта, рекомендуется использовать базу данных сигналов для хранения любых данных по проекту ПО, которые могут использоваться для проектирования, верификации и анализа прикладного ПО.

При разделении проекта прикладного ПО на подпрограммы (группы алгоритмов), как показано на рисунке 3, все связи между подпрограммами должны быть сформированы из переменных описанных в базе данных сигналов. В этом случае для каждой подпрограммы (рис. 3) векторы входных сигналов **i1\_in(t); i2\_in(t);…..iN\_in(t)** и векторы выходных сигналов **i1\_out(t); i2\_out(t); …. iN\_out(t)**, так же как и векторы входных воздействий **i\_in(t)** и выходных значений **Q\_out(t)** будут полностью описаны в проекте ПО.

Рекомендуется для каждой категории и группы сигналов вводить как минимум одно информационное поле, содержащее описание объекта (см. для примера поле Descript на рис.6).

## 5.5 Наименование переменных и система кодирования

При генерации кода из базы данных сигналов формируется имя переменной, состоящее из двух частей: «**имя группы сигналов**»\_«**имя сигнала**».

Например, для показания датчика, описанного в базе сигналов на рисунке 4, будет сформировано имя переменной **TE10P01B1\_XQ01,** где:

**TE10P01B1** – имя датчика в технологической схеме проекта;

**XQ01** – имя сигнала в категории датчики.

Для категорий, описывающих оборудование или элементы технологических схем АЭС, рекомендуется использовать, имена принятые для наименования оборудования и технологических схем в проекте.

В качестве имени группы сигналов для прикладного ПО, рекомендуется использовать код, созданный по системе кодирования, принятой для проектирования конкретной АЭС, например, систему кодирования ККS.

Система ККS, предназначенная для маркировки технологического оборудования, исполнительных органов (запорно-регулирующей, предохранительной, отсечной и т.п. арматуры, механизмов собственных нужд), точек измерения, монтажных единиц, устройств автоматизации, зданий и сооружений, система KKS позволяет маркировать алгоритмы и программы различного вида и назначения (алгоритмы обработки измеряемых технологических параметров, сигнализации, автоматического регулирования, технологических защит, логического управления: блокировок, АВР, пошаговых программ, — расчета технико-экономических показателей и диагностики состояния технологического оборудования), входные, выходные и промежуточные сигналы этих алгоритмов и программ, видеокадров всех уровней, отображаемые на видеотерминалах, кабели и пр.

## 5.6 Разграничение доступа к базе данных сигналов

База данных сигналов хранится в виде файла специального формата с расширением \*.db.

Для ограничения доступа и несанкционированного изменения, рекомендуется использовать системы разграничения прав доступа предоставляемые операционной системой и (или) информационными системами предприятия.

При групповой работе рекомендуется создавать главную базу данных под управлением системы хранения версий (SVN, Git).

Базы данных, разработанные отдельными участниками проекта, должны объединяться с главной базой данных с использованием процедуры слияния.

Средства сравнения баз данных SimInTech обеспечивают контроль за модификацией.

# **Проектирование расчетной схемы в SimInTech**

Для создания прикладной программы с помощью предметно ориентированного языка программирования SimInTech, необходимо в графическом виде создать функционально-блочную диаграмму. Диаграмма представляет собой графическое изображение преобразования входных величин в выходные путем математических вычислений. Для создания программы необходимо:

1. задать входные переменные;
2. задать выходные переменные;
3. создать схему алгоритмов, используя набор блоков SimInTech;
4. задать параметры блоков, включая начальные условия.

Каждая функционально-блочная диаграмма (расчетная схема) в SimInTech должна быть математически определенной и проверятся математическим моделированием в динамическом режиме. При задании любого вектора входных переменных, изменяющихся во времени, SimInTech обеспечивает расчет выходных переменных.

Любой проект прикладной программы в среде SimInTech состоит из набора отдельных функционально-блочных диаграмм (расчетных схем SimInTech).

Минимальной частью проекта является единичная функционально-блочная диаграмма – «***лист алгоритма***». Листы алгоритмов объединяются в «***группы алгоритмов***».

Каждая группа алгоритмов может содержать в себе как отдельные листы так другие группы алгоритмов (см. рис. 6).

Глубина вложенности структур проекта определяется требованиями проекта и разработчиком прикладного программного ПО.

В среде SimInTech возможна динамическая проверка каждого отдельного листа алгоритмов и любой группы алгоритмов, непосредственно во время проектирования.

Таким образом в общем виде любой проект прикладного ПО может быть представлен в виде древовидной структуры, состоящей из листов алгоритмов, объединенных в группы алгоритмов (см. рис. 5).

Рекомендуется формировать каждый лист алгоритма таким образом, чтобы он мог быть распечатан на листе стандартного формата, а весь проект при печати распечатывался бы как альбом диаграмм алгоритмов.



**Рисунок 6. Организация проекта прикладного ПО в SimInTech**

## Содержание расчетной схемы

## Оформление проекта прикладного ПО

Проектная расчетная схема должна содержать следующие атрибуты:

* Организация – название организации разработчика.
* Проект – название проекта.
* Система – название системы для которой создается прикладное ПО.
* Разработал – Фамилия и инициалы разработчика.
* Проверил – Фамилия и инициалы проверяющего.
* Утвердил – Фамилия и инициалы утверждающего.
* Ревизия – номер ревизии схемы.
* Дата ревизии – дата выпуска ревизии проекта ПО.

## Оформление листов алгоритмов

Лист алгоритма представляет собой субмодель, внутренняя структурная схема которой разделена на четыре области: область штампа; таблица входов; область логики алгоритма; таблица выходов.

Область штампа содержит атрибуты организации, исполнителя, проекта, технологической системы и т.д. В таблице входов указываются имена входных по отношению к данному листу сигналов, выбираемых из базы сигналов проекта.

В области логики реализуется структурная схема алгоритма, обеспечивающая необходимое преобразование входных сигналов листа алгоритма в его выходные сигналы.

В таблице выходов указываются имена выходных сигналов листа алгоритма. Содержание таблицы выходов автоматически переносится в соответствующую запись базы сигналов проекта (в таблицу «Алгоритмы», запись с полным именем листа алгоритма).

Основным атрибутом листа алгоритма является имя листа - уникальный в пределах алгоритма цифро-буквенный код. Субмодель «Лист алгоритма» вставляется в структурную схему субмодели «группа алгоритмов». Полное имя листа алгоритма складывается из имени алгоритма, которому он принадлежит, и имени листа.

Совокупность листов алгоритмов, имеющих одно общее имя, но разные имена листов, объединяются в понятие «Группа Алгоритм». На структурной схеме представляет собой субмодель, внутренняя структурная схема которой состоит из блоков типа «Лист алгоритма», либо «группа алгоритмов» Основным атрибутом алгоритма является его имя - уникальный в пределах всего комплексного проекта (проекта системы управления) цифро-буквенный код.

## Разграничение прав доступа к проекту

Расчетная схема хранится в виде файла специального формата, с расширением \*.prt

Для идентификации пользователя рекомендуется установить пароль доступа к проекту.

Для ограничения доступа в целях защиты от несанкционированного изменения, рекомендуется использовать системы разграничения прав доступа предоставляемые операционной системой и (или) информационными системами предприятия.

При групповой работе над проектом, необходимо осуществить разделение общей схемы между участниками, с использованием групп алгоритмов. Каждый участник проекта разрабатывает собственно группу алгоритмов из общего проекта. Каждая группа алгоритмов хранится в собственном файле.

Для каждой группы алгоритмов может быть установлен собственный пароль, обеспечивающий защиту от модификации.

Для каждого файла должны быть настроены права доступа и роли для всех участников проекта, с помощью системы разграничения прав доступа предоставляемой операционной системой и (или) информационными системами предприятия.

В зависимости от настроек информационной системы предприятия, может быть использован как ролевой так и индивидуальный доступ к проекту и его частям.

Рекомендуется создавать файлы проекта под управлением системы хранения версий. (SVN, GIT).

Все проекты, разработанные отдельными участниками, должны объединяться с главным проектом ответственным разработчиком.

Средства визуального сравнения проектов обеспечивают контроль за изменениями.

## Структура прикладного ПО на вычислительном приборе

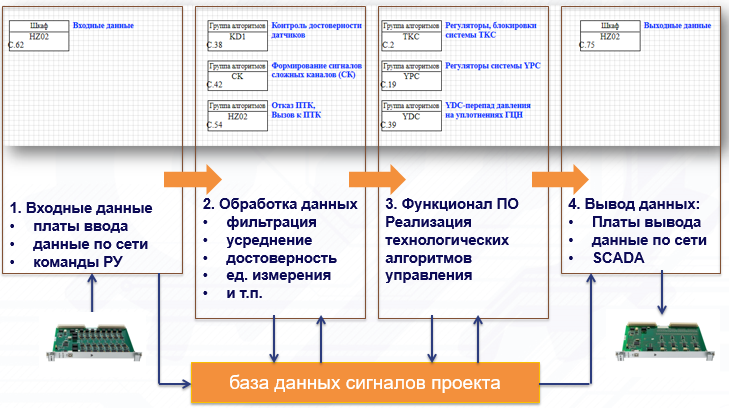
Проектирование прикладного ПО осуществляется «сверху вниз».

При проектировании прикладного ПО должен использоваться принцип «модульности», каждый отдельный лист алгоритмов и каждая группа алгоритмов может рассматриваться как отдельный модуль ПО, с возможностью изолированного тестирования и изменения.

Для каждого вычислительного узла, в SimInTech необходимо создать отдельный проект.

В общем случае, для создания проекта прикладного программного обеспечения для вычислительного узла SimInTech, рекомендуется разделять общий проект на верхнем уровне на следующие части, согласно последовательности обработки данных:

1. входные данные;
2. обработка данных;
3. функциональное программное обеспечение;
4. вывод данных (см. рис. 6).



**Рисунок 7. Структура верхнего уровня для прикладного ПО в SimInTech**

## Входные данные прикладного ПО

В этой части рекомендуется группировать алгоритмы, связанные с поступлением данных в программное обеспечение от внешних, по отношению к вычислительному прибору источников. Источниками могут быть:

* Платы ввода цифровые и аналоговые, подключенные к вычислителю.
* Интерфейсные платы с различными протоколами передачи данных.
* Вычислительные приборы, в объединённой сети.
* Команды оператора, полученные из системы управления верхнего уровня.

Для каждого типа источника рекомендуется создавать соответствующую категорию в базе данных сигналов SimInTech.

Алгоритмы данной части прикладного ПО отвечают за получение исходных данных и преобразование их в значения переменных в базе данных сигналов.

Рекомендуется добавлять в базу данных сигналов переменные, связанные с отказами источников сигналов для дальнейшей обработки в прикладном ПО.

Данная часть прикладного ПО может содержать функционал, связанный работой конкретных плат и зависит от используемой аппаратуры.

Рекомендуется использовать типовые решения для каждого отдельных определенных в проекте видов плат ввода и используемых протоколов сетевого соединения.

Рекомендуется в данной части прикладного ПО размещать все функции, связанные с взаимодействием с аппаратурой и системным программным обеспечением.

После выполнения данной части прикладного ПО на приборе в переменных программного обеспечения находятся актуальные значения, полученные от источников сигналов (показания датчиков, подключенных к платам ввода-вывода, сигналы об отказах, данные, полученные от системы верхнего уровня или по сети от других вычислительных приборов).

## Обработка данных

Данная часть прикладного ПО предназначена для первичной обработки данных, полученных от внешних источников. Здесь рекомендуется осуществлять необходимые подготовительные преобразования данных, такие как:

* Пересчет данных, от датчиков из вольт-амперных показаний в физические значения технологических параметров (Температура, Давление, Расход и т.п.).
* Оценку достоверности полученных данных (см. п. 6.2.2.1).
* Фильтрацию полученных данных с целью устранения возможных колебаний.
* Накопление буферов и счетчиков для последующих вычислений.
* Вычисление сложных сигналов, например, вычисление среднего значения по показаниям нескольких датчиков.
* Вычисление параметров, не измеряемых в прямую, по косвенным показателям, например, расчет расхода по перепаду давления.
* Логическая обработка резервируемых сигналов, выборка значений сигналов из набора для дальнейшей обработки, с учетом достоверности и данных по исправности оборудования.

После выполнения алгоритмов из этой группы в прикладном ПО будут обновлены значения всех переменных, описывающих состояние объекта управления. На стадии расчетного моделирования проекта переменные обновляются в базе данных сигналов SimInTech. Контролируемые параметры представлены в форматах и единицах измерения, заданных при проектировании ПО. В системе будет так же представлена информация об отказах в каналах контроля в виде значений соответствующих переменных (сигналов в базе данных).

Рекомендуется по возможности создавать типовые алгоритмы для обработки типовых сигналов, например, типовой алгоритм обработки сигнала с датчиков определенного типа.

## Самодиагностика ввода данных

На стадии обработки входных данных должна происходить проверка полученных сигналов на предмет их достоверности. В качестве критериев достоверности могут быть использованы:

* сравнение с максимальным и минимально возможным значением;
* оценка скорости изменения изменяемой величины;
* отсутствие изменений в течение заданного периода времени;
* отсутствии пакетов данных, полученных по сети;
* сравнение данных измерения одной и той же величины по двум и более каналам измерения.

Результаты работы диагностирующих алгоритмов записываются в общую базу данных сигналов и должны учитываться на следующем этапе обработки сигналов.

## Функциональное ПО

Данная группа алгоритмов прикладного ПО содержит описание функций управления, связанных с технологией работы контролируемой и управляемой системы. В этих алгоритмах на основании данных, подготовленных на предыдущей стадии, выполняется вычисление необходимого управляющего воздействия на управляемую систему. В зависимости от задачи системы управления здесь могут быть реализованы различные типы технологических алгоритмов управления, такие как:

* Регуляторы различного вида.
* Реализация защит по технологическим параметрам.
* Формирование предупредительной сигнализации.
* Функционально-групповое управление.
* Выполнение пошаговых команд управления.
* Выбор команд по приоритету.
* И т.п.

Основные особенности технологической части алгоритмов прикладного ПО:

* Разрабатываются на основе технологических требований к объекту управления. Рекомендуется разрабатывать непосредственно технологами в предметной области на стадии проектирования АЭС (см. п. 3.2.3.1).
* При разработке проекта систем АЭС, рекомендуется валидация алгоритмов путем математического моделирования алгоритмов в SimInTech совместно с математической моделью, выполненной с использованием аттестованных расчетных кодов.
* Не зависят от выбранной аппаратной платформы и системного программного обеспечения.

В результате выполнения данной части алгоритмов в базе данных сигналов будут заполнены значения переменных, готовых к выводу в СКУ.

## Самодиагностика функционального ПО

Алгоритмы должны учитывать достоверность полученных данных, результаты самодиагностики см. п. 6.2.2.1.

Данная часть алгоритмов должна содержать самодиагностику и алгоритмы на случай недостоверности входных данных.

В случае невозможности выработать управляющее воздействие из-за недостоверности входных данных, должны быть выработаны соответствующие команды, для передачи в систему верхнего уровня и оператору.

## Вывод данных 2

В этой части рекомендуется группировать алгоритмы, связанные передачей результатов работы прикладного ПО во внешние, по отношению к прикладному ПО приёмники сигналов.

Приёмниками могут быть:

* Платы вывода цифровые и аналоговые, подключенные процессору и передающие сигналы на органы управления.
* Интерфейсные платы с различными протоколами передачи данных.
* Вычислительные приборы, в объединённой сети.
* Системы сигнализации и отображения верхнего уровня управления.

Рекомендуется для типовых плат вывода создавать, отдельные листы алгоритмов и соответствующие структуры в базе данных сигналов.

Данная часть прикладного ПО может содержать функционал, связанный с работой конкретных плат и зависит от используемой аппаратуры.

Рекомендуется использовать типовые решения для каждого из отдельных определенных в проекте видов плат вывода и используемых протоколов сетевого соединения.

Рекомендуется в данной части прикладного ПО размещать все функции, связанные с взаимодействием с аппаратурой и системным программным обеспечением.

## Самодиагностика выходных данных

Рекомендуется осуществлять проверку выходных данных на предмет их достоверности.

Например:

Выход сигнала за заданные пределы.

Логическая совместимость команд. (исключение заведомо не не верных сочетаний).

Действительность рассчитанных выходных сигналов.

Должен быть предусмотрен вывод информации об отказах в систему верхнего уровня и (или) оператору.

# **Функциональная верификация и валидация**

Согласно требованиям стандарта ГОСТ МЭК 60880-2011 необходимо производить верификацию прикладного ПО в процессе разработки, независимо от средств разработки, используемых для создания ПО.

При использовании SimInTech процесс верификации может быть разделен на две части:

1. Верификация расчетной схемы прикладного ПО в SimInTech (модели) на соответствие требованиям проекта.
2. Верификация прикладного ПО уже установленного на оборудовании на предмет соответствия динамического поведения ПО и динамического поведения математической модели в SimInTech.

Во втором случае, осуществляется верификация, не только математической модели, но и средств компиляции ПО, а также системного программного обеспечения, установленного на компьютере.

Использование SimInTech обеспечивает повторное использование программных инструментов тестирования, созданных для проверки на уровне модели, для проверки на уровне готового оборудования.

Проект системы управления (расчётная схема) в SimInTech является математической моделью и верифицируется с помощью математического ядра.

Статический анализ схемы является обязательным этапом разработки и осуществляется автоматически при генерации исходного кода прикладного ПО. Автоматическая генерация кода не происходит, если статический анализ выявляет ошибки.

Динамический анализ расчетной схемы является рекомендуемой частью типового процесса создания прикладного ПО.

Рекомендуется проводить динамический анализ расчетной схемы путем математического моделирования динамики с различными векторами входных воздействий.

Рекомендуется использовать динамическую модель систем, созданную в рамках оценки безопасности АЭС, для совместного расчета объекта управления и системы управления в различных режимах работы.

## 7.1 Статический анализ проекта в SimInTech

Для выполнения статического анализа проекта прикладного ПО необходимо выполнить:

* проверку целостности базы данных;
* инициализацию расчетной схемы для динамического расчета или автоматическую генерацию исходного кода прикладного ПО.

## 7.1.1 Статический анализ базы данных сигналов.

Проверка целостности базы данных осуществляется средствам SimInTech и включает:

1. Проверку уникальности имен групп сигналов – имеются ли одинаковые имена групп сигналов в разных категориях.
2. Проверку уникальности имен сигналов внутри группы сигналов – имеются ли одинаковые имена сигналов внутри одной группы. Выполняется для всех групп всех категорий.
3. Проверку наличия русских букв в именах групп сигналов и сигналов – имеются ли русские буквы в именах сигналов или их групп. Выполняется для всех групп всех категорий и для всех сигналов всех групп.

Имена всех групп сигналов в пределах проекта должны быть уникальны, поскольку они используются для создания имени переменной в сгенерированном исходном коде прикладного ПО на универсальном языке.

Имена сигналов в пределах одной группы должны быть уникальны, поскольку данное имя используется для создания имени переменной в исходном коде на языке Си.

Наличие русских букв в именах групп сигналов и сигналов недопустимо, поскольку эти имена используются для формирования имени переменных при автоматической генерации кода.

## 7.1.2 Статический анализ расчетной схемы проекта

Статический анализ выполняется на стадии инициализации расчетной схемы.

В рамках статического анализа расчетной схемы математическое ядро осуществляет проверку следующих требований:

1. Использование допустимых для генерации кода расчетных блоков.
2. Топология расчетной схемы однозначно определена, все блоки соединены линиями связи.
3. Отсутствие алгебраических петель в схеме алгоритмов.
4. Все переменные расчетной схемы и свойства блоков однозначно определены.
5. Начальное состояние каждого блока и всей расчетной схемы определено.

В случае не выполнения условий 1 – 5 пользователь получает сообщение SimInTech c идентификацией ошибки и подсветкой блока, в котором произошла ошибка.

На этапе инициализации расчетной модели происходит сортировка блоков для определения порядка вычисления.

## 7.2 Динамический анализ расчетной схемы

Динамический анализ может проводиться для верификации как отдельных листов алгоритмов, так и групп алгоритмов (принцип модульности) в процессе разработки прикладного ПО.

Для динамического анализа расчетной схемы необходимо осуществить моделирование расчетной схемы прикладного ПО с заданием входных воздействий и анализом выходных сигналов во времени. Входные сигналы для модулей системы могут быть имитированы средствами SimInTech.

Для верификации прикладного ПО рекомендуется использовать комплексную модель, в которой расчетная схема прикладного ПО рассчитывается совместно с одним или несколькими тестовыми проектами, созданными в SimInTech. В комплексной модели используется единое модельное время и единая база сигналов проекта.

Для проекта прикладного ПО при динамическом моделировании необходимо использовать шаг моделирования, равный шагу дискретизации выполнения ПО на целевом компьютере.

Для обеспечения полноты тестирования по входным переменным динамическое тестирование должно обеспечивать подачу тестовых сигналов на все входные переменные, существующие в прикладном ПО. При необходимости создаются несколько тестовых проектов.

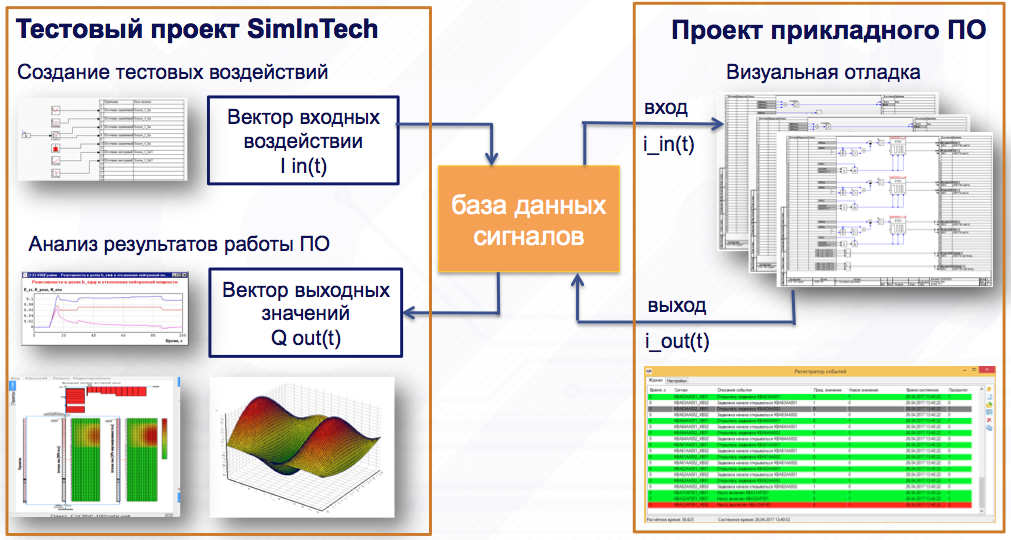
При моделировании различных режимов работы прикладного ПО, при необходимости, рекомендуется создавать несколько тестовых проектов, для формирования вектора входных сигналов, имитирующих работу объекта в различных режимах.

Последовательность действий при динамическом анализе расчетной схемы:

1. Создать тестовый проект или несколько проектов для формирования вектора входных воздействий и анализа результата работы ПО.
2. Создать тестовый пакет, в состав которого входит проект тестируемого прикладного ПО.
3. Установить начальное состояние для тестового пакета и тестируемого прикладного ПО.
4. Запустить тестовый пакет на расчет заданного отрезка модельного времени.
5. Осуществить фиксацию результатов работы проекта прикладного ПО.

## 7.2.1 Структура тестового пакета

Для реализации комплексной модели необходимо создать тестовых пакет SimInTech. Пакет SimInTech представляет собой несколько проектов SimInTech c единой базой данных сигналов. Рекомендованная структура текстового пакета изображена на рис. 8.



**Рисунок 8. Тестовый пакет проекта для тестирования прикладного ПО**

В общем случае тестовый пакет содержит:

* Базу сигналов Прикладного ПО.
* Один или несколько проектов прикладного ПО (расчетных схем SimInTech), предназначенных для тестирования.
* Один или несколько тестовых проектов SimInTech.

**База данных сигналов** содержит все сигналы проекта прикладного ПО, а также, при необходимости, дополнительные сигналы необходимые для тестирования и динамического анализа работы прикладного ПО. В процессе расчетного моделирования база данных отображает текущие значения сигналов (переменных) прикладного ПО и обеспечивает возможность «ручного ввода» изменения любых переменных проекта, для целей тестирования.

**Проект прикладного ПО** – созданная на предметно-ориентированном языке прикладная программа или ее часть, готовая для автоматической генерации кода.

В процессе расчетного моделирования графические средства SimInTech обеспечивают анимацию - отображение работы прикладного ПО. Пользователь может использовать следующие возможности анимации среды SimInTech для анализа работы ПО:

* Изменение цвета линий связи между блоками в зависимости от величины переменной, передающейся по линии связи.
* Просмотр значения переменных на линиях связи между блоками в диалоговых окнах.
* Наблюдение изменения параметров на анимированных изображениях расчетных блоков.
* Вывод величины сигнала.
* Создание регистратора событий для проекта прикладного ПО.

**Тестовый проект SimInTech** в пакете тестирования предназначен для создания изменяемого по времени вектора входных воздействий на проект прикладного ПО и анализа вектора выходных результатов работы.

Тестовый проект выполняется с использованием предметно-ориентированного языка программирования SimInTech.

Тестовый проект может содержать алгоритмы анализа вектора результата работы ПО. Варианты анализа результатов работы прикладного ПО в тестовом пакете:

* Сравнение с эталонными сигналами и вычисление расхождения.
* Анализ минимального и максимального значения сигнала.
* Время появления сигнала.
* Вычисление среднего значения сигнала.
* Счетчики для дискретных сигналов.
* Частотный анализ полученных сигналов.

Входные воздействия на проект прикладного ПО осуществляются через базу данных сигналов.

Выходные результаты в процессе моделирования переходного процесса отображаются в базе данных сигналов.

Графические средства анализа обеспечивают отображения полученных результатов в виде временных таблиц, графиков, включая фазовые портреты и трехмерные поверхности.

Для протоколирования процесса испытаний может быть использован журнал регистрации событий SimInTech.

Отладочная информация и результаты тестирования могут выводиться в окно сообщений SimInTech.

Для фиксации и повторного использования, результаты тестовых прогонов среды SimInTech рекомендуется записывать в файлы.

## 7.2.2 Варианты создания вектора тестового воздействия

Для создания тестового воздействия, необходимо в SimInTech создать отдельный расчетный проект, в котором будет происходить формирование входных сигналов алгоритмов. Источниками сигналов могут быть:

1. Тестовые циклограммы, последовательность изменения параметров.
2. Ручное изменение параметров, как в базе данных, так и с использованием визуального интерфейса.
3. Записанные в файл данные из математической модели объекта.
4. Математическая модель объекта, запущенная синхронно по модельному времени с расчетной схемой SimInTech.

Возможна комбинация способов задания тестового воздействия для обеспечения покрытия по входу.

## 7.2.2.1 Тестовые циклограммы

Для создания тестовых последовательностей входных сигналов в тестовом пакете могут быть использованы блоки из библиотеки «Источники» SimInTech.

Использование текстовых циклограмм обеспечивает создание детерминированных по времени входных воздействий на тестируемое ПО.

В рамках тестирования различных режимов работы могут должны быть созданы циклограммы, отвечающие за имитацию входных воздействий в различных режимах объекта.

Набор тестовых циклограмм должен покрывать все входные переменные прикладного ПО.

Для тестирования работы прикладного ПО в различных режимах работы необходимо создать набор тестовых циклограмм, обеспечивающих изменение входных параметров, которые соответствуют изменениям параметров в различных режимах работы объекта.

## 7.2.2.2 Ручное изменение входных параметров

Для воздействия на входные переменные в тестируемом прикладном ПО, можно осуществлять изменения переменных в «ручном режиме», для этого существуют следующие возможности в среде SimInTech:

* Изменение значений параметров в интерфейсе базы данных сигналов.
* Изменение параметров блоков на расчетной схеме тестового проекта, изменения сразу передаются в тестируемое ПО.
* Использование специальных интерфейсных блоков:
  + кнопка «вкл/выкл»;
  + блок разрыва линии связи;
  + блок ручного переключения.
* Создание графического интерфейса оператора, для ввода данных. Графическая подсистема SimInTech содержит средства создания графических интерфейсов оператора, которые обеспечивает создание мнемосхем и панелей управления.

При использовании ручного изменения параметров рекомендуется использовать моделирование в режиме синхронизации с реальным временем.

Рекомендуется осуществлять запись изменений, выполненных в ручном режиме для возможности повторения последовательности изменения в автоматическом режиме.

## 7.2.2.3 Данные моделирования из файла

Изменения входных переменных могут быть получены путем моделирования динамического объекта с использованием SimInTech и (или) других расчетных кодов.

Тестовая последовательность входных воздействий может быть записана в файл для воспроизведения в режиме моделирования.

Для различных режимов работы объекта может использоваться разные наборы предварительно записанных векторов входных сигналов.

## 7.2.2.4 Математическое моделирование систем АЭС

Частью любого проекта АЭС, является создание отчета по углубленной оценке безопасности, при его создании используется математическое моделирование динамики процессов, с помощью специализированных расчетных кодов.

Специализированные расчетные коды проходят аттестацию в государственных органах, осуществляющих регулирование в области ядреной энергетике.

Аттестованный расчетный код имеет установленную область применения для расчета физических процессов и определенную точность расчета.

SimInTech может быть использован для создания комплексной модели АЭС, включающей модель СКУ, созданную в SimInTech и модель системы АЭС.

Использование комплексной модели АЭС, созданной с помощью аттестованных расчетных кодов, обеспечивает верификацию прикладного ПО с приближением к условиям реальной работы ПО способом.

Математическая модель, созданная с использованием аттестованного расчетного кода, обеспечивает расчет контролируемых параметров систем АЭС с значениями, близкими к значениям параметров реального процесса на АЭС. Динамика изменения параметров также рассчитывается в соответствии с реальными процессами.

Для входных параметров прикладного ПО рекомендуется в среде SimInTech создавать математические модели, имитирующие работы реальных каналов измерения. Включая имитацию работы плат ввода-вывода.

Рассчитанные математической моделью физические величины, должны пересчитываться в входные переменные программы с учетом реальных физических каналов измерения и передачи данных. Должны быть учтены такие факторы искажения параметров как:

* инерционность датчиков;
* погрешность датчиков измерения;
* разрядность дискретизации;
* задержки в канале измерения;
* зашумление канала измерения.

Для выходных переменных прикладного ПО, осуществляющих воздействие на математическую модель должны быть созданы модели исполнительных механизмов, обеспечивающие моделирование реального воздействия прикладных программ на реальные системы АЭС.

Например, должно учитываться время запаздывания, связанное со временем открытия-закрытия арматуры, время разгона-торможения двигателей и т.п.

При необходимости должны быть использованы несколько моделей для тестирования всего вектора входных переменных в различных режимах работы управляемого объекта.

## 7.2.3 Анализ работы прикладного ПО в тестовом проекте

В процессе верификации должно быть осуществлено наблюдение и фиксация изменения выходных переменных программы при заданных входных воздействиях.

Для анализа выходных сигналов прикладного ПО необходимо в тестовой схеме SimInTech использовать блоки «чтение сигналов», которые обеспечивают передачу сигналов из базы сигналов на схему для анализа.

Для процесса верификации прикладного ПО, необходимо знать требуемое значение вектора выходных значений при заданных входных воздействиях.

Данные по требуемому значению вектора выходных воздействий должны быть подготовлены на стадии создания спецификации и требований к прикладному ПО.

Тестовый проект обеспечивает сравнение вектора выходных воздействии с требуемыми значениями.

Для дискретных выходных сигналов динамическим моделированием может быть оценено:

1. Совпадение значения выходного сигнала, с требуемым, при заданных входных сигналах (входные значения могут быть как аналоговыми, так и дискретными);
2. Время появления требуемого значения выходного сигнала, при заданных входных сигналах;
3. Время удержания требуемого значения выходного сигнала, при заданных входных сигналах.

Для аналоговых выходных сигналов динамическим анализом может быть оценено:

1. Совпадение значения выходного сигнала с требуемым, при заданных входных сигналах, с заданной точностью;
2. Отклонение значения выходного сигналов от требуемого, при заданных входных сигналах;
3. Время появления требуемого значения выходного сигнала, при заданных входных сигналах;
4. Время удержания требуемого значения выходного сигнала, при заданных входных сигналах.

При использовании математической модели в качестве значений верификации могут быть использованы рассчитываемые значения параметров технологического процесса, управляемого СКУ, для которой выполняется создание прикладного ПО.

## 7.2.3.1 Тестовое покрытие прикладного кода

При проведении динамического анализа прикладного ПО рекомендуется обеспечить тестовую проверку покрытия кода. Расчетная схема SimInTech обеспечивает возможность проверки изменения любых значений в процессе моделирования. Рекомендуется на стадии тестирования расчетной схемы прикладного ПО отслеживать изменения переменных, показывающих срабатывания участков кода.

## 7.2.3.1.1 Тестовое покрытие входных переменных

Тестовый проект должен содержать схему, проверяющую изменение входных значений в процессе выполнения моделирования. Для каждого тестового режима необходимо создавать список входных переменных, которые изменились в процессе моделирования.

Необходимо обеспечить такое количество тестов, при котором каждая входная переменная менялась в процессе моделирования хотя бы в одном из тестов.

## 7.2.3.1.2 Тестовое покрытие выходных переменных

Тестовый проект должен содержать схему, проверяющую изменение выходных значений прикладного ПО в процессе моделирования. Для каждого тестового режима, необходимо создавать список выходных переменных, которые изменились в процессе моделирования.

Необходимо обеспечить такое количество тестов, при котором каждая выходная переменная менялась в процессе моделирования хотя бы в одном из тестов.

## 7.2.3.1.3 Тестовое покрытие ветвления и логических переключений

Для блоков SimInTech, обеспечивающих переключение расчетной схемы, необходимо обеспечить проверку срабатывания каждого из вариантов переключения. Блоки, переключающие состояние схемы, находятся в библиотеках «Ключи», «Логические», «Нелинейные», «Триггеры», «Релейные».

Для проверки переключения необходимо на линии связи соответствующих блоков установить специальные блоки контроля изменений. Для каждого теста необходимо создавать список блоков, для которых произошло переключение.

Необходимо обеспечить такое количество тестов, при котором для каждого блока переключения схемы переключение происходило хотя бы в одном из тестов.

# **Генерация кода прикладного ПО**

Для создания исходного кода на универсальном языке программирования используется модуль генерации кода SimInTech.

В качестве исходных данных для генерации кода используется:

1) Проект прикладного программного обеспечения в среде SimInTech, созданный согласно рекомендаций п. 4.1.

2) Шаблон исходного кода на универсальном языке, настроенный для конкретного системного программного обеспечения и технических средств.

В результате выполнения генерации кода создается исходный код прикладного ПО, полностью готовый к компиляции и загрузке на выбранные технические средства.

В случае генерации кода на языке Си, файлы с исходными текстами прикладного программного обеспечения создаются и хранятся отдельно от файлов шаблона для генерации кода. Данные файлы могут быть использованы для анализа исходного кода сторонними средствами статического анализа.

Код, созданный средой SimInTech, может быть интегрирован в любые другие программы на целевой системе.

## 8.1 Настройка генерации кода в SimInTech

Среда SimInTech должна быть настроена таким образом, чтобы была обеспечена связь с средствами компиляции и отладки объектного кода на выбранных технических средствах.

Настройки системы должны обеспечивать автоматический вызов программного обеспечения компиляции и создания объектного кода.

В случае использования SimInTech на инструментальной рабочей станции рекомендуется настраивать связь SimInTech c техническими средствами аппаратуры управления таким образом, чтобы созданный объектный код автоматически передавался на технические средства.

На технических средствах рекомендуется использовать специальные средства отладки, обеспечивающие запуск прикладного программного обеспечения в режиме «оn line» отладки с отображением работы прикладного ПО на расчетной схеме SimInTech.

## 8.2 Верификация технических средств системы контроля и управления

Рекомендуется использовать для проверки прикладного ПО, установленного на компьютеры системы контроля и управления тестовые проекты, созданные для проверки проекта прикладного ПО на этапе функциональной верификации (см. п. 7.2).

Тестовый проект может быть подключен к компьютеру системы контроля и управления, работающему совместно с SimInTech в режиме отладки. Тестовый пакет запускается в режиме синхронизации с реальным временем. В этом случае производится тестирование работы функциональной части программного обеспечения, работающего на компьютере. Данный режим обеспечивает проверку работы прикладного ПО, без учета работы плат ввода-вывода.

Данная проверка позволяет оценить правильность работы программы на компьютере системы контроля и управления.

Сравнение выходного вектора результатов работы, полученного на компьютере СКУ и вектора, полученного при динамическом моделировании расчетной схемы, позволяет проверить работу компилятора и системного программного обеспечения и выявить возможные отклонения и дефекты в компиляторе и системном программном обеспечении.

## 8.2.1 Полигон тестирования СКУ

Для полноценного тестирования СКУ тестовый проект может быть подключен к системе испытаний, которая осуществляет аппаратное преобразование данных тестового проекта в токовые сигналы, соответствующие сигналам с реальных датчиков системы контроля и управления.

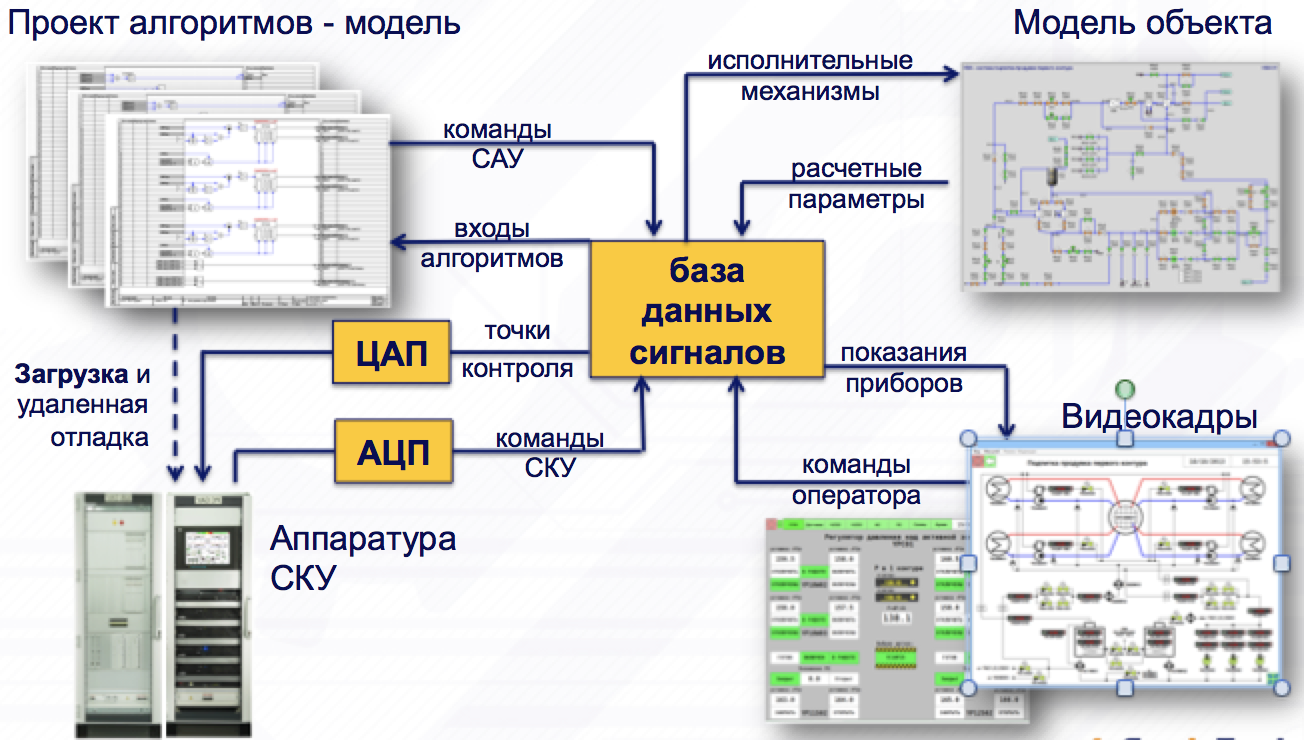
В этом случае кроме проверки функциональной части программного обеспечения, осуществляется проверка плат ввода-вывода СКУ, а также интерфейсных плат и интерфейсов связи с системами верхнего уровня.

Тестовые проекты, созданные на этапе функциональной верификации ПО, могут быть использованы для создания тестовых воздействий на технические средства на этапе тестирования на полигоне.

В случае использования комплексных математических моделей АЭС (см. п. 7.2.2.4) возможно полномасштабное моделирование различных режимов работы АЭС, для которых созданы математические модели, включая режимы нормальной эксплуатации, переходные и аварийные режимы работы АЭС.

Для целей тестирования СКУ необходимо обеспечить запуск моделирования комплексной модели АЭС в режиме синхронизации с реальным временем.

В общем случае схема проверки технических средств с использованием тестовых проектов представлена на рис. 9.



**Рисунок 9. Схема тестирования СКУ с использованием комплексной модели SimInTech**

# **Соответствие требованиям ГОСТ Р МЭК 60880-2011 и типового процесса разработки прикладного ПО в среде SimInTech**

|  |  |  |
| --- | --- | --- |
| **Требования ГОСТ Р МЭК 60880-2011** | | **Типовой процесс проектирования в SimInTech** |
| 5.4 Управление проектированием ПО  5.4.1 Любое проектирование ПО должно быть разбито на несколько этапов. | | 4.1 Этапы разработки ПО |
| 5.4.6 Когда деятельность по разработке программного обеспечения автоматизируется с использованием инструментальных программ, эта автоматизированная деятельность должна быть документально оформлена, включая документацию по входным и выходным данным, относящимся к этапу. | | 7.2 Динамический анализ расчетной схемы  7.2.1 Структура текстового пакета |
| 5.4.7 Входные и выходные данные на каждом этапе должны быть определены и документально оформлены. | | 2.3 Структура база данных  3.1 Оформление листов алгоритмов |
| 5.4.9 Каждый этап должен включать создание соответствующих документов. | | 4.1 Этапы разработки прикладного ПО в SimInTech |
| 5.5.13 Каждый шаг верификации или анализа должен заканчиваться отчетом о проведенном анализе, сделанных выводах и согласованных решениях. Отчет должен быть включен в документацию. | | 7.2 Динамический анализ расчетной схемы |
| 5.6 Управление конфигурацией.  5.6.4 Каждая создаваемая версия любого программного продукта, должна иметь уникальную идентификацию. | | 4.1.1 Проектирование расчетной схемы в SimInTech |
| 5.6.5 Должна иметься возможность идентификации всей документации связанной с программный продуктом. | | 3.1.1 Оформление проекта расчетной схемы SimInTech  6.1.2 Оформление листов алгоритмов |
| 5.6.7 Должна иметься возможность идентификации версий всех программных продуктов, которые совместно образуют полную версию конечного программного продукта. | | 6.1 Содержание проекта прикладного ПО  6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| 5.6.11 Доступ к программному обеспечению должен быть ограничен | | 2.5 Разграничение доступа к базе данных сигналов.  3.1.3 Разграничение прав доступа к проекту |
| 5.7.3 Доступ пользователя | | 2.5 Разграничение доступа к базе данных сигналов  3.1.3 Разграничение прав доступа к проекту. |
| 6.1.3 В требованиях к программному обеспечению должны указываться:   * –  прикладные функции, обеспечиваемые программным обеспечением; * –  различные типы поведения программного обеспечения и соответствующие  условия перехода; * –  интерфейсы и взаимодействия программного обеспечения со средой  (например, с операторами, с остальными элементами СКУ, с другими системами, если они существуют, с которыми оно взаимодействует или разделяет ресурсы), включая роли, типы, форматы, диапазоны и ограничения на вводы и выводы данных; * –  параметры программного обеспечения, которые могут быть модифицированы вручную во время операции, если таковые существуют, их роли, типы, форматы, диапазоны и ограничения, а также проверки, которые должно осуществлять программное обеспечение, когда эти параметры изменяются; | | 5.2 Спецификация требований к прикладному ПО в SimInTech  5.4.1 Рекомендации по созданию структуры базы данных сигналов |
| 6.4.3 Спецификация требований к программному обеспечению должна быть однозначной, тестируемой или верифицируемой, а также достижимой. | | 5.2 Спецификация требований к прикладному ПО в SimInTech |
| 6.2 Самоконтроль | | 6.2.1 Входные данные прикладного ПО  6.2.2 Обработка входных данных  6.2.2.1 Самодиагностика входных данных прикладного ПО  6.2.3 Функциональное ПО |
| 6.4 Документация  6.4.2 Спецификация требований должна быть представлена в стандартизированном формате, который не должен влиять на понятность документа. | | 6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| 7.1.1.3 Структура программного обеспечения должна основываться на модульном принципе. | | 6.2 Структура прикладного ПО |
| 7.1.1.1 Проект программного обеспечения должен содержать самоконтроль. | | 6.2.1 Входные данные прикладного ПО  6.2.2 Обработка входных данных  6.2.2.1 Самодиагностика входных данных прикладного ПО  6.2.3 Функциональное ПО |
| 7.1.1.7 Исходная программа должна соответствовать оформленным документально правилам, предназначенным для улучшения ясности, модифицируемости и тестируемости. | | 6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| 7.1.3 Реализация нового программного обеспечения на проблемно-ориентированных языках | | 6 Проектирование схемы в SimInTech |
| 7.1.3.1 Рекомендуется, чтобы методы формализации обладали следующими свойствами: невысокая сложность, ясность и стандартность расположения и представления, модульность, наличие соответствующих комментариев, отсутствие небезопасных элементов. Эти свойства, как правило, упрощают понимание, верификацию, тестирование и последующую модификацию. | | 6 Проектирование схемы в SimInTech |
| 7.1.3.4 Рекомендуется, чтобы проблемно-ориентированные языки позволяли разработчикам учитывать спецификацию проекта архитектуры СКУ, например, давали возможность назначать функции компонентам системы и поддерживали способность проекта сохранять устойчивость к дефектам элементов технического обеспечения. | | 6.2.2 Ввод данных6.2.2.1 Самодиагностика ввода данных 6.2.2.4 Вывод данных |
| 7.2.3 Проблемно-ориентированные языки и соответствующая автоматизированная генерация кода | | 8 Генерация кода прикладного ПО. |
| 7.4.1 Во время разработки программного обеспечения этап проектирования должен заканчиваться составлением спецификации проекта программного обеспечения. | | 6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| 7.4.2 Документ должен быть структурирован в соответствии с уровнями процесса проектирования программного обеспечения.  Спецификация проекта программного обеспечения может быть представлена в виде одного документа или полного набора отдельных документов. | | 6.2 Структура прикладного ПО |
| 8 Верификация программного обеспечения  8.1.15 Верификация программного обеспечения  8.2.3.2.1 Рекомендуется, чтобы прикладное программное обеспечение, которое автоматически генерируется из спецификации, использующей проблемно- ориентированный язык, имело систематизированную структуру с целью поддержания эффективной верификации. | | 4.1 Этапы разработки ПО  5.2 Спецификация требований к прикладному ПО  7 Функциональная верификация и валидация |
| 8.2.3.2.2 Программное обеспечение, написанное на проблемно- ориентированных языках, должно верифицироваться на правильность и согласованность с помощью, например, визуальной проверки или с использованием автоматических инструментальных программ, которые позволяют моделировать работу программного обеспечения в режиме отладки. | | 5.2 Спецификация требований к прикладному ПО  6 Проектирование расчетной схемы в SimInTech  7.2 Динамический анализ расчетной схемы |
| 9.3 Верификация интегрированной системы | | 8.2 Верификация технических средств контроля и управления  8.2.1 Полигон тестирования СКУ |
| 10.2 Валидация системы  10.2.1 Система должна проверяться с помощью статического и динамического моделирования входных сигналов, существующих при нормальной эксплуатации, при ожидаемых эксплуатационных событиях и при аварийной ситуации, требующей реакции испытываемой компьютерной системы. | | 8.2.1 Полигон тестирования СКУ |
| 14 Инструментальные программы для разработки программного обеспечения  14.1 Общие сведения | | 1 Введение |
| 14.3.6 Автоматизация тестирования | | 7.2 Динамический анализ расчетной схемы |
| 14.3.6.1Рекомендуется, чтобы инструментальные программы автоматизированной валидации, которые вырабатывают испытательные данные, передают или преобразуют эти данные и результаты тестирований, а также оценивают результаты тестирований, осуществляли полное составление протокола. Это применимо как к тестированиям модулей, так и к моделированиям станции. | | 7.2.1 Структура тестового пакета  7.2.3 Анализ работы прикладного ПО в тестовом проекта |
| 14.3.6.2 Соответствующие инструментальные программы следует использовать при тестированиях и/или моделировании поведения рабочей программы, загружаемой в целевую систему. | | 7.2.2 Варианты создания вектора тестового воздействия  8.2.1 Полигон тестирования СКУ |
| ПРИЛОЖЕНИЕ А Жизненный цикл безопасности программного обеспечения и детализация требований к программному обеспечению | | |
| A1 Жизненный цикл | | 4.1 Этапы разработки прикладного ПО |
| A.2.2 Самоконтроль | | 6.2.2.1 Самодиагностика ввода данных  6.2.3.1 Самодиагностика функционального программного обеспечения  6.2.4.1 Самодиагностика выходных данных |
| А.2.3 Представление спецификаций программного обеспечения  А.2.3.1 Спецификации программного обеспечения должны быть легко понимаемыми всеми группами пользователей. | | 5.2 Спецификация требований к прикладному ПО  6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| ПРИЛОЖНИЕ В Детализированные требования и рекомендации по проектированию и реализации | | |
| B1a Модифицируемость  В1.а Процесс проектирования должен сделать ПО легко модифицируемым. | | 4.2 Рекомендуемая структура прикладного ПО  5.4 Структура базы данных сигналов |
| B1b Подход «сверху вниз»  B1.b При проектировании следует использовать подход сверху вниз. | | 5.4 Структура базы данных сигналов  6.2 Структура прикладного ПО на вычислительном приборе |
| B1.bb На каждом этапе проектирования всю систему следует полностью описать и подвергнуть верификации | | 4.1 Этапы разработки прикладного ПО |
| B1.bh Следует проводить проектирование и разработку ПО, используя одно или несколько формализованных описаний высокого уровня (где это целесообразно и эффективно), подобно тому, как это делается в математической логике, теории множеств, а также использовать псевдокод, таблицы решений, логические схемы, другие графические средства или проблемно-ориентированные языки | | 1 Введение  6 Проектирование расчетной схемы в SimInTech |
| B1.bi Следует использовать автоматические средства разработки | | 1 Введение  8 Генерация кода прикладного ПО |
| B1.c Верификация промежуточных результатов разработка | | 4.2 Рекомендуемая структура прикладного ПО  7 Функциональная верификация и валидация |
| B1.ca Промежуточные результаты должны верифицироваться | | 4.1 Этапы разработки ПО  4.2 Рекомендуемая структура прикладного ПО  7 Функциональная верификация и валидация |
| B1.d Управление модификацией в процессе разработки | | 5.6 Разграничение доступа к базе данных.  6.1.3 Разграничение прав доступа к проекту |
| B2 Структура программного обеспечения  B2.a Программы и их части должны систематически группироваться | | 6.2 Рекомендуемая структура прикладного ПО |
| B2.aа Рекомендуется, чтобы специальные операции системы выполнялись специальными частями | | 6.2 Рекомендуемая структура прикладного ПО |
| В1.ab ПО следует разбивать на части так, чтобы аспекты, связанные с такими функциями, как:  - внешние интерфейсы компьютера (например, управление устройствами, обработка прерываний);  - сигналы реального времени (например, часы);  - параллельная обработка (например, блок оперативного управления);  - размещение памяти;  - специальные функции (например, утилиты);  - размещение стандартных функций на технических средствах конкретного компьютера;  были отделены от прикладных программ с хорошо определенными интерфейсами между ними | | 6.2 Рекомендуемая структура прикладного ПО  6.2.1 Входные данные прикладного ПО  6.2.2 Обработка данных  6.2.3 Функциональное ПО  6.2.4 Вывод данных |
| В1.ad Следует четко формулировать используемые методы структурирования | | 6.2 Рекомендуемая структура прикладного ПО |
| В1.af Однозначная и четкая модульная структура компьютерной программы | | 6.2 Рекомендуемая структура прикладного ПО |
| B3. Самоконтроль  B3.a Проверки достоверности  B3.ab Следует проверять области изменения:  - входных переменных;  - выходных переменных;  - промежуточных параметров, включая проверку границы массива | | 6.2.2.1 Самодиагностика ввода данных |
| B.4 Детальное проектирование и кодирование  B4.е Структуры данных  Структуры данных и соглашения о наименовании данных должны использоваться в единообразной форме по всей системе | | 5.4.1 Рекомендации по созданию структуры базы данных |
| B4.еа. Переменные, массивы и ячейки памяти должны иметь единственное назначение и структуру. Следует избегать использования эквивалентности | | 5.4.1 Рекомендации по созданию структуры базы данных  5.5 Наименование переменных и система кодирования |
| B4.еd. Когда в качестве глобальной структуры используется «база данных» с возможностью универсального доступа или аналогичные ресурсы, то доступ к ним должен обеспечиваться через стандартные процедуры обработки ресурсов или через связь со стандартными задачами манипулирования ресурсами. | | 5.4.1 Рекомендации по созданию структуры базы данных |
| B4.g Тестирование модулей и комплексные тестирования должны проводиться во время разработки программы | | 4.1 Этапы разработки ПО  7 Функциональная верификация и валидация |
| B4.gc Следует составлять формализованное описание входных тестовых данных и результатов (протокол тестирования) | | 7.2.2 Варианты создания вектора тестового воздействия |
| B5 Рекомендации зависящие от языка  B5.a Необходимо разработать подробные правила оформления различных языковых конструкций | | 6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| B5.d Правила кодирования | | 5.5 Наименования переменных и система кодирования  6.1.1 Оформление проекта прикладного ПО  6.1.2 Оформление листов алгоритмов |
| B5.e Проблемно-ориентированные языки  (нормативное)  Использование проблемно-ориентированных языков предпочтительней использования машинно-ориентированных языков | | 1 Введение  4 Типовой процесс проектирования в SimInTech |
| Приложение С Примеры технологии прикладного программирования, (разработка программного обеспечения с использованием проблемно-ориентированных языков) | | |
| С.1 Принципы применения требований | | |
| * С.1 а) структурированный по этапам процесс разработки программного обеспечения, с четким определением входной и выходной информации | 4.1 Этапы разработки прикладного ПО | |
| * С.1 b) понятная структура программного обеспечения, разрабатываемая на этапе проектирования программного обеспечения, которая формирует основу для  кодирования, а также для оценки программного обеспечения | 6.2 Структура прикладного ПО на вычислительном приборе | |
| * С.1 с) правила кодирования и технология, соответствующие требованиям и рекомендациям приведенными в приложении В. | 6 Проектирование расчетной схемы в SimInTech | |
| C.2 Применение требований к жизненному циклу программного обеспечения | 4.1 Этапы разработки ПО | |
| С.3 Применение требований к автоматической генерации кода | 6 Проектирование расчетной схемы SimInTech | |
| C.3.1 Инструментальные программы для управления проектированием и разработкой | 4 Типовой процесс проектирования в SimIntech | |
| C.3.2 Инструментальные программы автоматической генерации | 4.1 Этапы разработки ПО | |
| Приложение Е Верификация и тестирование программного обеспечения | | |
| Е.1 Деятельность по верификации и тестированию программного обеспечения | 7 Функциональная верификация и валидация | |
| Е.2 Системный подход | 7.1 Статический анализ проекта SimInTech  7.2 Динамический анализ расчетной схемы | |